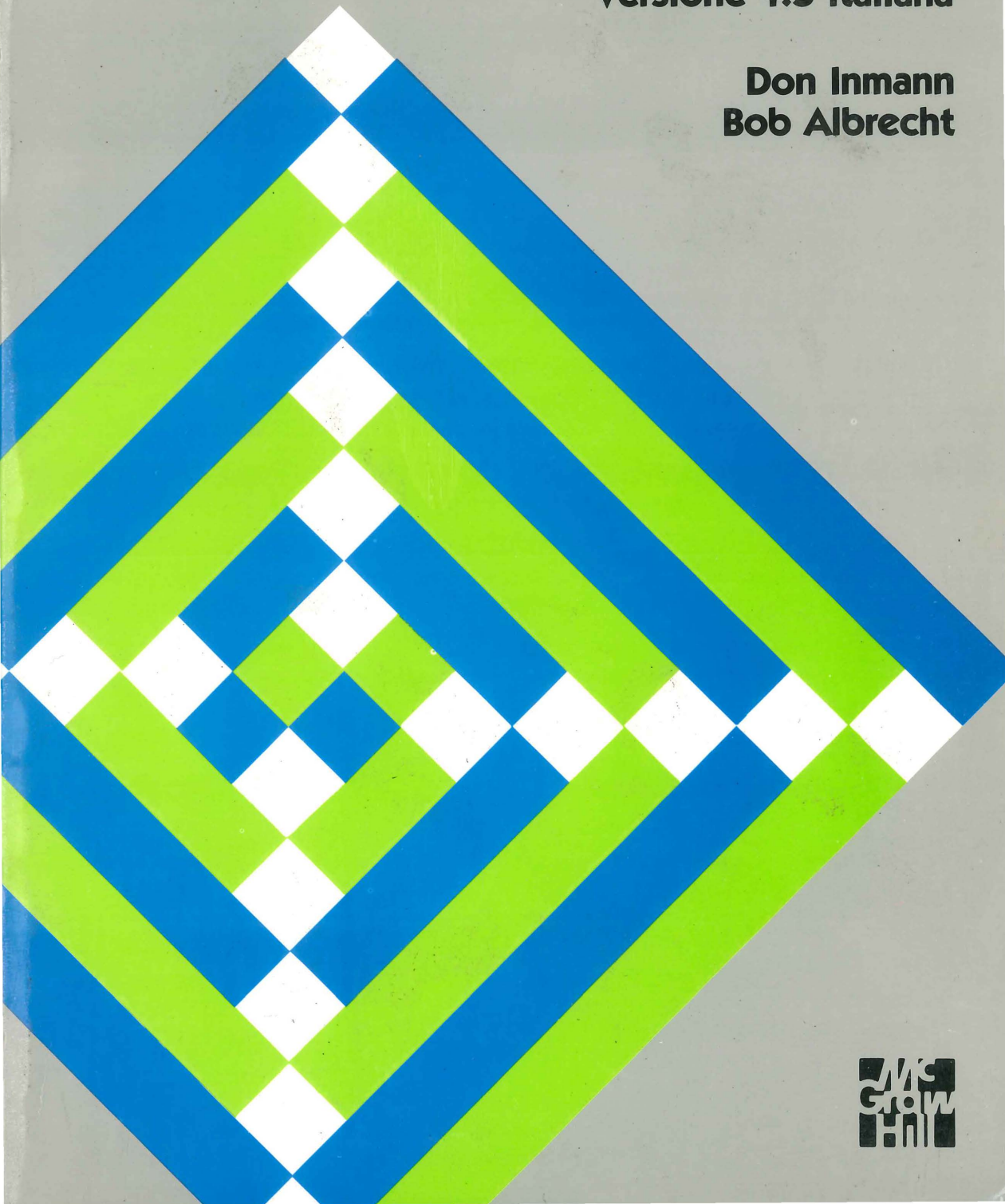
Programmare in QuickBASIC

Versione 4.5 italiana

Don Inmann
Bob Albrecht



McGraw Hill

Programmare in QuickBASIC

Versione 4.5 italiana

Dal catalogo McGraw-Hill:

The Waite Group's, *MS-DOS QBasic guida del programmatore*
Cornell, *Come usare Visual Basic 2*
Jamsa, *Guida rapida MS-DOS QBasic*
Ainsbury, *I segreti del DOS (con 2 floppy disk)*
Livingston, *I segreti di Windows 3.1 (con 3 floppy disk)*
Hoffman, *Come usare Word 6 per DOS*
Soucie, *Come usare Microsoft Excel 4 per Windows*
Davis, *Come usare Microsoft Access*

Don Inmann
Bob Albrecht

Programmare in QuickBASIC

Versione 4.5 italiana

McGRAW-HILL Libri Italia srl

Milano • New York • St. Louis • San Francisco • Oklahoma City
Auckland • Bogotá • Caracas • Hamburg • Lisboa
London • Madrid • Montreal • New Delhi • Paris • San Juan
São Paulo • Singapore • Sydney • Tokyo • Toronto

Ogni cura è stata posta nella verifica e nella documentazione delle informazioni contenute in questo libro, tuttavia né gli Autori né la McGraw-Hill Libri Italia possono assumersi alcuna responsabilità derivante dal loro utilizzo; lo stesso dicasi per ogni persona o società coinvolta nella creazione, nella produzione e nella distribuzione di questo libro.

Titolo originale: Using QuickBASIC 4.5, Second Edition

Copyright© 1988, 1989 McGraw-Hill, Inc.

Copyright© 1991 McGraw-Hill Libri Italia srl
Piazza Emilia 5
20129 Milano

I diritti di traduzione, di riproduzione, di memorizzazione elettronica e di adattamento totale e parziale con qualsiasi mezzo (compresi i microfilm e le copie fotostatiche) sono riservati per tutti i paesi.

Realizzazione editoriale: COMOD, Via Desiderio 21, 20131 Milano
Sede di Milano -Via Sangallo 41, 20133 Milano
Sede di Imperia -Via Generale Garibaldi 27/29, 18010 Cipressa

Copertina: Marco Mazzucato

Stampa: Arti Grafiche Battaia Snc - Rozzano (MI)

ISBN 88-386-0172-0

1 ª edizione marzo 1991

Printed in Italy

4567890GBTLLC9654321

Apple e Apple IIGS sono marchi registrati da Apple Computer Inc.; Atari è un marchio registrato da Atari Inc.; Microsoft, MS-DOS e GW-BASIC sono marchi registrati da Microsoft Corp; HP è un marchio registrato daHewlett-Packard Company; IBM è un marchio registrato da International Business Machines Corporation; Tandy è un marchio registrato daTandy Corporation.

Indice

Introduzione XIII

Capitolo 1 - Il BASIC 1

1.1 QuickBASIC: il meglio di due mondi 4
1.2 Approccio comune al BASIC 5
1.3 Esempi 6
1.3.1 Controllo della bolletta 6
1.3.2 Calcolo del valore dei titoli 10
1.4 Tecniche di programmazione 12
1.4.1 Convenzioni e stili 13
1.4.1.1 Istruzioni REM 14
1.4.1.2 Parole chiave BASIC 15
1.4.1.3 Variabili 15
1.4.1.4 Numerazione delle linee 15
1.4.1.5 Convenzioni di punteggiatura 16
1.4.1.6 Simboli di operazione e di relazione 16
1.4.1.7 Indentazione dei cicli 16
1.4.1.8 Istruzioni multiple 17
1.4.2 Convenzioni utilizzate 17
1.5 QuickBASIC e GW-BASIC 20
1.5.1 Comandi e funzioni proibite 21
1.5.2 Comandi che richiedono una modifica 22
1.5.3 Anteprima di QuickBASIC 23

1.6 Sommario 24

Capitolo 2 - Introduzione a QuickBASIC 4.5 25

2.1 Copie di sicurezza 26
2.2 Prime nozioni 27
2.3 Realizzazione di un programma QuickBASIC 28
2.3.1 Ambiente schermo di QuickBASIC 30
2.3.2 Creazione del programma LANCIO MONETE 31
2.3.3 Memorizzazione del programma LANCIO MONETE 32
2.3.4 Esecuzione del programma LANCIO MONETE 35
2.4 Utilizzo di un programma QuickBASIC già memorizzato 38
2.4.1 Richiamo di un programma 39
2.4.2 Esecuzione di un programma 40
2.5 Utilizzo di un programma GW-BASIC già memorizzato 41
2.5.1 Richiamo di un programma GW-BASIC 41
2.5.2 Memorizzazione di un programma GW-BASIC 42
2.5.3 Esecuzione di un programma GW-BASIC 43
2.6 Altri Menu 44
2.6.1 Menu Modifica 44
2.6.2 Menu Visualizza 44
2.6.3 Menu Ricerca 45
2.6.4 Menu Debug 45
2.6.5 Menu Opzioni 45
2.6.6 Menu Guida 46
2.7 Sommario 46

Capitolo 3 - Realizzazione di strumenti di lavoro 47

3.1 Creazione di un insieme di strumenti 48
3.2 Variabili e tipi di dati 48
3.3 Linee di programma 51

3.4 Funzioni predefinite 52
3.5 Funzioni definite dall'utente 53
3.6 Dischetto strumenti 55
3.7 Definizione di funzioni 56
3.7.1 Funzioni Massimo e Minimo 56
3.7.2 Definizione di funzioni senza parametri 58
3.7.3 Utilizzo di definizioni di funzione 58
3.8 Funzioni su più linee 62
3.9 Strutture di controllo 66
3.9.1 Blocco IF...THEN...ELSE 66
3.9.2 Struttura SELECT CASE 69
3.9.3 Struttura DO...LOOP 75
3.10 Sommario 78

Capitolo 4 - Editing 79

4.1 Menu completo 79
4.2 Spostamenti in un file 81
4.2.1 Spostamento al principio ed al termine di un file 82
4.2.2 Spostamento una linea verso l'alto o verso il basso 83
4.2.3 Spostamento al principio ed alla fine di una linea 83
4.2.4 Spostamento a sinistra ed a destra di una parola 84
4.2.5 Spostamento a destra ed a sinistra di un carattere 85
4.2.6 Scorrimento dello schermo verso l'alto o verso il basso di una finestra . . .85
4.2.7 Scorrimento di una linea verso l'alto o verso il basso 86
4.2.8 Scorrimento di una finestra verso destra o verso sinistra 87
4.3 Inserimento e cancellazione di testo 87
4.3.1 Inizio di una nuova linea 88
4.3.2 Inserimento di testo 88
4.3.3 Cancellazione di testo 93
4.4. Comandi del menu Modifica 94

4.4.1 Taglia, Annulla, Copia ed Incolla 95
4.4.1.1 Selezione del testo 95
4.4.1.2 Taglia 97
4.4.1.3 Annulla 97
4.4.1.4 Taglia ed Incolla 99
4.4.1.5 Copia 100
4.4.1.6 Cancella 101
4.4.1.7 Altri comandi del menu Modifica 101
4.4.2 Inserimento di file 102
4.5 Sommario 105

Capitolo 5 - Procedure 107

5.1 Procedure FUNCTION 107
5.2 Procedure SUB 114
5.3 Array 117
5.3.1 Dimensionamento degli array 117
5.3.2 Passaggio di array a procedure FUNCTION 120
5.4 Passaggio di un array ad una procedura SUB 124
5.5 Utilizzo di più procedure 128
5.6 Visualizzazione dei sottoprogrammi 133
5.7 Sommario 135

Capitolo 6 - File sequenziali non strutturati 137

6.1 Tipi di file di dati 138
6.2 Attribuzione di un nome ad un file 141
6.3 Creazione di un file sequenziale 142
6.3.1 Programma per la creazione del file di esempio 144
6.3.2 Creazione del file 146
6.4 Lettura 150
6.4.1 Programma di lettura di file sequenziali 151

6.5 Stampa 153
6.5.1 Programma di stampa di file sequenziali 154
6.6 Aggiunte al file 156
6.6.1 Programma di aggiunta al file sequenziale 156
6.7 Programma integrato per il file d'esempio 158
6.7.1 Assemblaggio delle singole parti 160
6.7.2 Inclusione di CREA UN FILE in NOTES CON MENU 161
6.8 Sommario 168

Capitolo 7 - File sequenziali strutturati 169

7.1 Stati ed abbreviazioni 169
7.2 Ricerca di informazioni nel file di esempio 178
7.3 Memorizzazione delle informazioni 183
7.3.1 Modifica del programma 186
7.4 Sommario 191

Capitolo 8 - File ad accesso casuale 193

8.1 Fondamenti 194
8.2 Creazione di un file ad accesso casuale 194
8.3 Lettura di un file ad accesso casuale 200
8.4 Esempio: valore nutrizionale degli alimenti 204
8.4.1 Sottoprogramma AggiungiRecord 208
8.4.2 Sottoprogramma LeggiRecord 212
8.4.3 Scelta di un record per numero 213
8.4.4 Ricerca dei record in base al nome dell'alimento 218
8.4.5 Caso della scelta errata 221
8.5 Sommario 224

Capitolo 9 - Gestione dei file 227

9.1 Copia di un file sequenziale 227

9.1.1 Aggiunta del sottoprogramma NomeDrive 230
9.1.2 Aggiunta del sottoprogramma CopiaFile 232
9.1.3 Aggiunta del sottoprogramma EsaminaFile 234
9.2 Copia e modifica di un file sequenziale 239
9.2.1 Aggiunta dei sottoprogrammi NuovoDrive ed EsaminaFile 240
9.2.2 Aggiunta del sottoprogramma CopiaModifica 241
9.2.3 Aggiunta del Sottoprogramma AggiungiFile 242
9.2.4 Utilizzo del programma COPIA E MODIFICA UN FILE SEQUENZIALE 243
9.3 Ordinamento di un file ad accesso casuale 250
9.3.1 Aggiunta del sottoprogramma NomeFile 252
9.3.2 Aggiunta del sottoprogramma MescolaFile 253
9.3.3 Aggiunta del sottoprogramma LeggiRecord 254
9.3.4 Aggiunta del sottoprogramma OrdinaFile 255
9.4 Sommario 259

Capitolo 10 - Funzioni dinamiche di debug 261

10.1 Menu Debug 262
10.2 Analisi del flusso di un programma 263
10.2.1 Analisi del flusso 264
10.2.2 Esecuzione passo a passo 267
10.2.3 Esecuzione per procedure 268
10.3 Osservazione di un programma 270
10.3.1 Punto di osservazione ed Osserva 272
10.3.2 Punto di interruzione ed Osservazione immediata 277
10.4 Altri strumenti di debug 279
10.4.1 Resoconto 280
10.4.2 Interrompi su errori 280
10.4.3 Imposta l'istruzione successiva 280
10.5 Sommario 281

Capitolo 11 - File eseguibili 283

11.1 Creazione di file eseguibili 283
11.1.1 Preparazione dei dischetti di lavoro 284
11.1.2 Scelta dei percorsi 286
11.1.3 Meccanismo di creazione dei file .EXE 286
11.2 Creazione di un file .EXE che richiede BRUN45.EXE 290
11.2.1 Programma GRAFICI A BARRE 291
11.2.2 Creazione del file .EXE 294
11.2.3 Esecuzione del file .EXE con BRUN45.EXE 296
11.3 Creazione di un file .EXE autonomo 297
11.3.1 Programma GRAFICI A TORTA 298
11.3.2 Creazione del file autonomo 309
11.3.3 Esecuzione del file autonomo 310
11.4 Sommario 310

Capitolo 12 - Librerie Quick 311

12.1 Vantaggi delle librerie Quick 311
12.2 Creazione di una libreria Quick 312
12.2.1 File necessari 312
12.2.2 Libreria Quick creata da QuickBASIC313 . . . . . .
12.2.3 Dimostrazione della creazione di una libreria Quick 316
12.3 Libreria Quick UTIL1 319
12.4 Aggiunte ad un libreria Quick 320
12.4.1 Prova della libreria Quick UTIL2 324
12.4.2 Ulteriori aggiunte 327
12.4.3 Prova della libreria Quick UTIL3 328
12.5 Sommario 331

Appendice A - Parole riservate 333

Introduzione

Il BASIC è stato per lungo tempo il liguaggio per PC più usato e molti utenti hanno imparato a programmare proprio in BASIC per la sua semplicità d'apprendimento e d'uso, portando nel corso degli anni alla nascita di molti dialetti dall'iniziale semplice versione.

QuickBASIC Microsoft rappresenta un notevole balzo in avanti rispetto alle altre versioni, divenendo un linguaggio realmente strutturato, potente e professionale. Dato che trae le sue radici dal Microsoft BASIC, ne ha mantenuto la facilità di apprendimento e d'uso ed inoltre la maggior parte dei programmi che sono stati scritti con questo linguaggio rimangono compatibili, anche se possono essere facilmente aggiornati per sfruttare tutte le nuove potenti prestazioni del QuickBASIC.

In QuickBASIC 4.5 è possibile editare, compilare, lanciare ed effettuare il debug in un unico ambiente e tutti gli strumenti necessari od utili sono contenuti nel pacchetto.

Nel presentare l'argomento si sono date per scontate le conoscenze di base non solo del sistema operativo, ma anche della programmazione, del linguaggio ed una certa esperienza.

Il testo contiene molti esempi pratici e programmi per semplificare l'apprendimento delle procedure, delle istruzioni e delle funzioni QuickBASIC; il testo è supportato da una grande quantità di riproduzioni relative a quanto avviene sullo schermo, passo a passo, il tutto in un ambiente di facile lettura.

Lo stile di programmazione presentato è chiaro, lineare, strutturato e consistente: sono stati intenzionalmente evitati quei trucchi e quelle scorciatoie che avrebbero potuto distrarre dall'apprendimento.

Le capacità personali di programmazione sono sviluppate dallo studio dei programmi e dagli esperimenti di applicazione utilizzando gli elementi del linguaggio: questo libro non può che incoraggiare a seguire questa via, rendendo il tutto meno tedioso e più proficuo.

Capitolo 1

Il BASIC

1.1 QuickBASIC: il meglio di due mondi
1.2 Approccio comune al BASIC
1.3 Esempi
1.4 Tecniche di programmazione
1.5 QuickBASIC e GW-BASIC
1.6 Sommario

Il BASIC (Beginner's All-purpuse Symbolic Instruction Code) era disponibile, quale linguaggio di programmazione per grandi calcolatori, molto prima che apparissero i primi microcalcolatori; prese origine dal desiderio di rendere disponibile la potenza di calcolo non solo ai programmatori professionisti che disponevano di grandi e costosi calcolatori, ma a chiunque.

I professori John G. Kemeny e Thomas E. Kurtz dall'univeristà di Dartmouth svilupparono il linguaggio BASIC originale nel 1963 e 1964 come uno strumento didattico per l'insegnamento a programmatori inesperti che fosse facile da apprendere, ma allo stesso tempo adatto a qualsiasi necessità di programmazione. Il successo ed il notevole sviluppo del BASIC sono dovuti alla sua semplicità, facilità d'impiego e versatilità.

Nel 1965 fu disponibile anche al di fuori di Dartmouth: inizialmente attraverso sistemi time-sharing (a ripartizione dei tempo) ed in seguito anche su minicalcolatori dedicati a basso prezzo (relativamente a quei tempi).

Quando vennero offerti i microcalcolatori al vasto pubblico come personal computer (elaboratori personali), i calcolatori si trasformarono da esclusivo possesso dei programmatori professionisti a quello degli utenti ed amatori ed il solo linguaggio di alto livello disponibile per quelle prime macchine era il BASIC.

Segue una breve storia del BASIC.

1955 Viene progettata una versione semplificata del linguaggio di programmazione assembler, Darsimco (Dartmouth Simplified Code), ma non prende piede.

1956-57 Kemeny e Kurtz divengono sempre più consci della carenza di programmi di batch-processing.

1957 Compare il FORTRAN per esperti.

Primi '60 Kemeny e Kurtz incominciano a prendere seriamente in considerazione il time-sharing per l'utilizzo di calcolatori da parte di più persone; la National Science Foundation approva il finanziamento ad un progetto, portato avanti da Kemeny, Kurtz e da una dozzina di studenti, volto allo sviluppo di un sistema time-sharing al Dartmouth College, contrassegnando così l'avvento del calcolo interattivo personalizzato: Kemeny, Kurtz e gli studenti lavorano con compilatori FORTRAN, ALGOL e linguaggi sperimentali che influenzerano in seguito l'evolvere del BASIC.

1963 Dartmouth decide di operare una alfabetizzazione informatica di tutti gli studenti e quindi Kemeny e Kurtz, con l'aiuto di studenti, intraprendono la messa a punto di un sistema time-sharing di uso generale e di un nuovo linguaggio (BASIC) per presentare la programmazione ai principianti, adatto ad applicazioni in piccoli e grandi sistemi.

Si stabiliscono le caratteristiche originali del BASIC:

- essere di uso generale, adatto alla stesura di qualsiasi tipo di programma;
- permettere l'eventuale introduzione a posteriori di nuove caratteristiche;
- permettere l'interazione utente-calcolatore;
- fornire messaggi di errore chiari e cortesi;
- dare un responso veloce a brevi programmi;
- non richiedere la conoscenza dell'hardware;
- mascherare all'utente il sistema operativo.

1964 L'equipaggiamento necessario allo sviluppo del sistema time-sharing del BASIC giunge nel febbraio e per il primo di marzo è completamente funzionante; il primo maggio alle ore quattro del mattino John Kemeny ed uno studente programmatore entrano nel sistema ed eseguono contemporaneamente e con successo due programmi BASIC distinti. Nascono così il time-sharing ed il BASIC.

1964-1971 L'evoluzione del BASIC in questo periodo precede i personal computer.

A Dartmouth il BASIC, da linguaggio adatto solamente a brevi programmi, diviene un linguaggio idoneo per importanti programmi

applicativi (Kemeny e Kurtz ebbero unicamente responsabilità per il primo BASIC di Dartmouth; altri si interessarono in seguito).

Nacquero due rami genealogici principali dal BASIC originale di Dartmouth: le versioni di BASIC per grandi calcolatori, di probabile discendenza diretta dal GE BASIC, a sua volta discendente del BASIC di Dartmouth, e le versioni per piccoli calcolatori, discendenti di versioni apparse su Hewlett Packard HP-2000 o Digital PDP-8. Il BASIC di Dartmouth ed il GE BASIC sono stati quasi identici sino a circa il 1970: Dartmouth ha infatti sviluppato il sistema time-sharing con hardware GE.

L'estensione delle istruzioni PRINT e l'introduzione dell'istruzione INPUT, avvenuta quasi due anni dopo la nascita del BASIC, migliora notevolmente l'interazione tra utente e calcolatore.

Si introduce l'uso di file (archivi) anche per scopi differenti dalla memorizzazione di programmi.

La possibilità di chiamare subroutine (sottoprogrammi) esterne accresce la flessibilità di lunghi programmi.

Alla fine del 1971 Il BASIC di Dartmouth ha raggiunto la sua sesta versione ed un immenso successo.

Le prime versioni di BASIC erano, per la limitata memoria di cui disponevano i minicalcolatori di quei tempi, dei precari adattamenti di quello di Dartmouth, mentre oggi ne è disponibile almeno una versione per ogni marca di microcalcolatore; queste, pur essendo più complete e potenti, sono ancora semplici da apprendere ed utilizzare. Il BASIC è attualmente parte integrante od accessoria di quasi ogni calcolatore negli uffici, case e scuole. Il BASIC, attraverso comandi che provocano una immediata risposta allo schermo, permette una stretta interazione con l'utente, incoraggiandolo così con risultati istantanei; allo stesso tempo il desiderio di risolvere problemi di particolare interesse o di ottenere schermate più gradevoli, stimola il desiderio e contemporaneamente la necessità di apprendere ulteriori capacità del linguaggio.

Questa immediata e costante interazione tra uomo e calcolatore pone il BASIC in prima linea tra i linguaggi di programmazione rivolti al vasto pubblico. Sono ora disponibili molti linguaggi per microcalcolatore: COBOL, FORTRAN, LOGO, Pascal, Modula-2 e C, per nominarne alcuni, e la lista si allunga costantemente; ognuno ha il suo gruppo di sostenitori che manifesta i vantaggi della propria scelta. Il BASIC non è un linguaggio standardizzato, ne esistono infatti versioni differenti adatte all'hardware di ogni singolo calcolatore ed anche gli stili di programmazione degli utenti sono molto vari; vengono spesso redatti programmi funzionanti, ma comprensibili solamente al programmatore stesso che, a sua volta, potrebbe avere difficoltà nel rileggerlo a posteriori. I sostenitori di altri linguaggi criticano il BASIC perché non si presta ad una programmazione ben strutturata, ma, con un po' di pianificazione e sforzo da parte del programmatore, è possibile strutturare il programma in blocchi esercitanti singole funzioni; inoltre gli ultimi interpreti e compilatori

BASIC, come il QuikBASIC, si qualificano come linguaggi strutturati: con funzioni su più linee, procedure con variabili locali e globali e strutture come IF...THEN...ELSE...END IF e DO...LOOP. Alcune versioni del BASIC, QuickBASIC incluso, hanno ora tutte le buone caratteristiche di cui i sostenitori di altri linguaggi si erano vantati in passato.

Il BASIC è anche stato criticato per la sua lentezza; questo era vero nel passato, dal momento che la maggior parte delle versioni erano di tipo interpretato. Il microprocessore di un calcolatore non è in grado di eseguire direttamente le istruzioni BASIC; può solamente eseguire istruzioni di codice macchina, rendendo quindi necessaria, prima della esecuzione, una traduzione nel linguaggio macchina del processore. Esistono due metodi per operare la traduzione: con un interprete e con un compilatore. La quasi totalità delle versioni sviluppate nel passato, e molte delle odierne, sono interpretate; ogni istruzione è quindi tradotta (interpretata) e poi eseguita, rallentando così il processo di esecuzione del programma con questa interpretazione linea per linea che ha luogo ogniqualvolta il programma viene eseguito.

La versione originale del BASIC, e la sue successive evoluzioni a Dartmouth, facevano uso di un compilatore per la conversione dei programmi BASIC in linguaggio macchina; un compilatore traduce (compila) le istruzioni BASIC prima dell'esecuzione del programma che, compilato, produce un file eseguibile in linguaggio macchina; quest'ultimo, quando viene eseguito, gira molto velocemente dal momento che si tratta del linguaggio proprio della macchina e la traduzione ha già avuto luogo, evitando così spreco di tempo in questa operazione durante l'esecuzione. Gran parte del software in commercio opera in questo modo.

Vi sono vantaggi e svantaggi in entrambe i metodi di traduzione; i programmi compilati sovente richiedono più tempo per l'eliminazione degli errori poiché, quando viene scoperto un errore nel file eseguibile in linguaggio macchina, il programma originale in BASIC (detto codice sorgente) deve essere corretto e ricompilato prima che il suo file eseguibile (detto codice oggetto) possa essere nuovamente lanciato. Se vengono rilevati altri errori, il programma deve essere corretto e ricompilato di nuovo. Benché il codice compilato venga eseguito molto velocemente, può essere necessario un tempo assai lungo per avere un programma privo di errori, mentre il BASIC interpretato, essendo tradotto linea per linea, permette una loro individuazione e correzione rapida; tuttavia la velocità di esecuzione è rallentata rispetto a quella di un programma compilato.

1.1 QuickBASIC: il meglio di due mondi

QuickBASIC 4.5 è l'ultima versione del Quickbasic della Microsoft Corporation e, pur mantenendo la caratteristica interattività che gli utenti del BASIC hanno imparato ad amare, permette un'esecuzione dei programmi molto più rapida rispetto alle versioni di BASIC interpretato. Ideato per IBM PC e PC compatibili, il QuickBASIC 4.5 offre funzioni e velocità precedentemente irreperibili con altre versioni di BASIC e combina l'interattività,

punto di forza del BASIC, con l'approccio strutturato, modulare, tipico dei linguaggi come il Pascal.

QuickBASIC fa uso del medesimo ambiente, potente, ma facile da utilizzare, caratteristico delle altre versioni di Microsoft BASIC; è compatibile con lo Advanced BASIC della IBM (BASICA) e con il GW-BASIC Microsoft che funziona su PC-compatibili. Il compilatore QuickBASIC è rivolto a qualsiasi tipo di utente e chi è familiare con Microsoft BASIC troverà semplice la transizione a QuickBASIC.

Di seguito vengono riportate alcune delle principali caratteristiche del QuickBASIC.

- È compatibile con Microsoft GW-BASIC e IBM BASICA.
- Offre un editor (redattore) a tutto schermo incorporato, un compilatore, un debugger (correttore) con menu a tendina e riquadri di comunicazione. Molti errori vengono rilevati dall'editor non appena si introduce l'istruzione.
- Contiene QB Advisor, un testo elettronico di riferimento, per un aiuto rapido e di facile accesso.
- Possiede una interfaccia con l'utente nuova e di facile uso, con Easy menu (menu facili) per principianti e Full menu (menu completi) per i più esperti.
- Possiede QB Express, una lezione in tempo reale per avviare i principianti. Esiste anche una lezione passo a passo che guida il principiante in una applicazione completa.
- Permette la selezione dei comandi con mouse o da tastiera.
- Assiste alla programmazione strutturata per mezzo di etichette alfanumeriche, istruzioni logiche strutturate, sottoprogrammi e funzioni su più linee, rendendo più agevole la lettura e la comprensione dei programmi.
- Permette l'uso della grafica (inclusi i modi CGA, EGA, e VGA), di BLOAD, BSAVE, musica ed effetti.

1.2 Approccio comune al BASIC

Poiché i lettori di questo libro ricoprono probabilmente un'ampia gamma di esperienze nel settore, è utile una breve rassegna della serie di istruzioni, funzioni e comandi BASIC ordinari. Si presuppone che il lettore abbia avuto modo di utilizzare una o più versioni di BASIC, indifferentemente Atari, Commodore, IBM PC, Tandy 1000, Apple II od altro: tutte le versioni di BASIC adottate da questi calcolatori sono simili.

Questa rassegna, per varie ragioni, è limitata a GW-BASIC (o IBM BASICA, che è quasi identica) e Applesoft BASIC: Microsoft BASIC è stata scelta perché è una delle più recenti e potenti versioni del BASIC, largamente impiegata e disponibile per molte marche di calcolatori, mentre la Applesoft BASIC (che è licenza della Microsoft) è stata scelta perché

è il linguaggio di una intera serie di calcolatori Apple II. Applesoft è un BASIC di vecchia data, installato su Apple II, II+, IIc IIe, ed anche sul nuovo IIGS. Per semplificare la trattazione, ci si riferirà in seguito a queste versioni di BASIC semplicemente con GW-BASIC ed Applesoft.

Queste versioni più o meno recenti, permettono con le loro similitudini e differenze di evidenziare un buon contrasto che illustra l'evoluzione del BASIC dalla vecchia e primitiva versione di Applesoft alla più recente e potente di GW-BASIC. La serie di brevi programmi presentati metterà in evidenza le similitudini e le differenze tra GW-BASIC e Applesoft.

I programmi Applesoft inclusi in questo capitolo faciliteranno la transizione da Applesoft a GW-BASIC e QuickBASIC 4.5: questi utenti potranno capire, confrontando le due versioni di ogni programma, come modificare rapidamente e con facilità i programmi Applesoft in modo che possano essere compilati ed eseguiti in QuickBASIC. Le versioni Applesoft e GW-BASIC sono state rese il più simile possibile in modo da rendere facile l'individuazione di analogie e differenze, ma naturalmente nel corso del libro si imparerà ad utilizzare le caratteristiche di QuickBASIC che non sono disponibili né in Applesoft né in GW-BASIC.

Molti dei lettori desidereranno solo leggere velocemente i semplici programmi di questa rassegna, studiando solamente quelli di particolare interesse, ma, nel caso siano principianti, dovrebbero studiare ogni programma e provarlo sul proprio calcolatore.

Le istruzioni REMARK (REM) vengono tralasciate quando il programma va in esecuzione, hanno il solo scopo di agevolare la lettura e la comprensione del programma: si può dunque risparmiare tempo omettendo di digitare queste istruzioni nel provare i programmi dimostrativi. Gli utenti BASIC esperti potranno evitare la lettura di questo capitolo e procedere con gli argomenti più avanzati.

1.3 Esempi

Il BASIC è essenzialmente un linguaggio di uso personale piuttosto che commerciale e quindi, i programmi sono applicazioni di uso comune, sono scritti in modo da essere facilmente leggibili e comprensibili: si sono suddivise le linee che adempiono a funzioni distinte in blocchi, ognuno dei quali è accompagnato da una istruzione REMARK.

1.3.1 Controllo della bolletta

Il programma BOLLETTA (GW-BASIC) controlla la correttezza della bolletta per luce e gas ed è proposto in due versioni: una in GW-BASIC e l'altra in Applesoft.

Il Programma 1.1, BOLLETTA, è scritto in GW-BASIC, che è compatibile con QuickBASIC, ed è quindi possibile compilarlo ed eseguirlo con QuickBASIC.

Anche i programmi in BASICA sono compatibili con QuickBASIC.

Il Programma 1.2, BOLLETTA (Applesoft), è scritto in Applesoft, che non è compatibile con QuickBASIC e non si potrà pertanto utilizzare QuickBASIC per la sua compilazione ed esecuzione. I Programmi 1.1 e 1.2 illustrano lo stile e molte delle convenzioni adoperate nel corso di questo testo. I programmi sono strutturati in blocchi funzionali, o moduli, ognuno dei quali ha inizio con una istruzione REMARK che identifica la funzione adempiuta; nelle linee che iniziano con un'istruzione REM non vengono mai utilizzate istruzioni GOTO, GOSUB od altre, in modo che sarà possibile omettere di digitarle: il programma funzionerà indifferentemente con o senza queste linee. Le istruzioni REM, che identificano i blocchi funzionali del programma, possono essere viste come un abbozzo del programma stesso. Ad esempio, quello che segue è un abbozzo dei Programmi 1.1 e 1.2 con le istruzioni REM (i valori dei prezzi sono esclusivamente di esempio):

```
1     REM  ** BOLLETTA **                          Blocco 1
100   REM  ** ASSEGNAMENTO COSTANTI **             Blocco 100
200   REM  ** IMMISSIONE TOTALI **                 Blocco 200
300   REM  ** CALCOLO BOLLETTA **                  Blocco 300
400   REM  ** ARROTONDA & STAMPA RISULTATI **      Blocco 400
```

```
1  REM ** BOLLETTA **
2  ' Capitolo 1
3  ' GW-BASIC File: UQBO101.BAS
4  ' Eventuali commenti
5  ' aggiuntivi al programma
9  '
100 REM ** ASSEGNAMENTO COSTANTI **
110 KWP = .0705            'PREZZO DELL'ELETTRICITA PER KWH
120 THP = .505             'PREZZO DEL GAS PER MC
130 TXR = .00125           'SOVRATASSA PER ENERGIA
140 ST$ = "L.#.###.###"
199 '
200 REM ** IMMISSIONE TOTALI **
210 CLS: KEY OFF
220 INPUT "TOTALE   KWH"; KWH
230 INPUT "TOTALE   MC"; THM
299 '
300 REM **CALCOLO BOLLRTTA **
310 ELECT = KWH * KWP
320 GAS = THM * THP
399 '
400 REM ** ARROTONDA & STAMPA RISULTATI **
410 ELECTOT = INT(ELECT * 100 + .5) / 100
420 GASTOT = INT(GAS * 100 + .5) / 100
430 PRINT : PRINT "TOTALE ELETTRICITA = ";: PRINT USING ST$; ELECTOT
440 PRINT "TOTALE GAS           = ";: PRINT USING ST$; GASTOT
```

```
450 SUBTOTAL = ELECTOT + GASTOT
460 TAX = INT(TXR * SUBTOTAL * 100 + .5) / 100
470 PRINT "SOVRATASSA ENERGIA       = ";: PRINT USING ST$; TAX
480 TOTAL = SUBTOTAL + TAX
490 PRINT: PRINT "BOLLETTA TOTALE   = ";: PRINT USING ST$; TOTAL
500 END
```

Programma 1.1 BOLLETTA GW-BASIC

Si osservi che il blocco di apertura (Blocco 1) è composto da sole istruzioni REM utilizzate per identificare questo particolare programma: nome, titolo del libro, capitolo, linguaggio utilizzato e nome del file. In questo gruppo di linee potranno di volta in volta essere incluse altre informazioni quali la data, commenti al programma, istruzioni per l'uso od altro.

Nello stendere un programma, si adoperino a proprio piacere le istruzioni REM, ma ci si assicuri di farlo: in questo modo il programma illustrerà se stesso, il che è particolarmente importante nel caso di lunghi programmi compilati, consentendo ad altri di leggere e comprendere i programmi.

```
1  REM ** BOLLETTA **
2  'Capitolo 1
3  ' Basic Applesoft  File: UQB0102.BAS
100 REM ** ASSEGNAMENTO COSTANTI **
110 KWP = .0705            'PREZZO DELLA ELETTRICITA PER KWH
120 THP = .505             'PREZZO DEL GAS PER MC
130 TXR = .00125           'RATEO DELLA TASSA PER ENERGIA
199 :
200 REM ** IMMISSIONE TOTALI **
210 HOME
220 INPUT "TOTALE   KWH ? "; KWH
230 INPUT "TOTALE   MC ? "; THM
299 :
300 REM **CALCOLO BOLLETTA **
310 ELECT = KWH * KWP
320 GAS = THM * THP
399 :
400 REM ** ARROTONDA & STAMPA RISULTATI **
410 TELEC = INT(ELECT * 100 + .5) / 100
420 TGAS = INT(GAS * 100 + .5) / 100
430 PRINT : PRINT "TOTALE ELETTRICITA = L. "; TELEC
440 PRINT "TOTALE GAS         = L. "; TGAS
450 SUBTTL = TELEC + TGAS
460 TAX = INT(TXR * SUBTTL * 100 + .5) / 100
470 PRINT "TASSA ENERGIA      = L. 0"; TAX
480 TT = SUBTTL + TAX
490 PRINT: PRINT "BOLLETTA TOTALE    = L. "; TT
500 END
```

Programma 1.2 BOLLETTA-Applesoft

GW-BASIC permette di utilizzare tanto l'apostrofo (') quanto REM per indicare l'istruzione REMARK; è anche possibile utilizzare solamente l'apostrofo in una linea di programma per separare i blocchi, come nelle linee 9, 199, 299 e 399 della versione in GW-BASIC di BOLLETTA; per adempiere alla stessa funzione in Applesoft si utilizzano i due punti (:). È anche possibile aggiungere commenti al termine delle linee del programma in GW-BASIC, come nelle linee 110, 120, 130, e 140 del Programma 1.1; in Applesoft vengono utilizzati i due punti, come nelle linee 110, 120 e 130 del Programma 1.2.

Il secondo blocco (blocco 100) assegna i valori dei prezzi unitari di gas ed elettricità ed il rateo di imposta sull'energia. Nella versione GW-BASIC viene adoperata un'altra variabile, ST$, per determinare il formato di stampa delle cifre in valuta, mentre in Applesoft, non essendo disponibile questa possibilità, è necessario digitare gli spazi in ogni singola istruzione PRINT.

Il terzo blocco (blocco 300) permette di inserire le unità di gas ed elettricità utilizzate in un determinato periodo; la pulizia dello schermo viene portata a termine con istruzioni differenti nei due programmi: in GW-BASIC e QuickBASIC con CLS, mentre in Applesoft con HOME, che pulisce lo schermo e porta il cursore nell'angolo in alto a sinistra.

È possibile osservare lievi differenze anche nelle istruzioni INPUT utilizzate per l'introduzione dei dati: in GW-BASIC viene offerta la scelta di porre o meno un punto di domanda al termine del messaggio di richiesta dati, mentre in Applesoft ne è sempre obbligatorio l'uso.

Il quarto blocco (blocco 300) calcola il costo del gas e dell'elettricità relativamente al periodo in esame ed è identico in entrambe le versioni.

Il quinto blocco (blocco 400) arrotonda i risultati del blocco 300, stampa questi valori arrotondati, calcola l'imposta sull'energia, la stampa, somma i risultati e stampa l'addebito totale. Alcune delle variabili in questo blocco differiscono nelle due versioni, infatti una delle più apprezzabili differenze tra GW-BASIC e Applesoft è proprio nell'utilizzo dei nomi di variabili: GW-BASIC e QuickBASIC accettano nomi di variabili sino a 40 caratteri, mentre Applesoft accetta nomi di non più di 8 caratteri. Inoltre, Applesoft riconosce solamente i primi due caratteri sulla sinistra del nome di una variabile, rendendo indispensabile molta attenzione nel caso si utilizzassero nomi più lunghi di due sole lettere.

Entrambe le versioni fanno uso della variabile ELECT per rappresentare il costo dell'elettricità, quella in GW-BASIC utilizza in seguito alla linea 410 ELECTOT per il valore arrotondato di ELECT, ma Applesoft, poiché riconosce solamente i primi due caratteri (EL), le considera come se fossero la stessa variabile.

Esiste un'ulteriore e più importante ragione per cui non è possibile utilizzare ELECTOT come variabile in Applesoft: mentre GW-BASIC preserva le indentazioni e le spaziature adoperate, Applesoft rimuove le indentazioni ed utilizza proprie convenzioni di spaziatura nel listare il programma. Infatti la spaziatura imposta da Applesoft potrebbe avere come risultato un inaspettato utilizzo di parole riservate; si supponga ad esempio di aver inserito la linea 410 nella versione Applesoft come segue:

```
410 ELECTOT = INT(ELECT * 100 + .5)/ 100
```

Sebbene non sembri apparire alcuna parola riservata in questa linea, questa, quando listata, verrà stampata:

```
ELEC TO T = INT(ELECT * 100 + .5) / 100
```

Applesoft trova la parola riservata TO, la separa con degli spazi e crea una linea priva di significato. La versione corretta Applesoft di BOLLETTA utilizza il nome TELEC anziché ELECTOT per la variabile; altri nomi di variabili che hanno dovuto essere modificati nel programma Applesoft sono: GASTOT (modificato in TGAS), SUBTOTAL (modificato in SUBTTL) e TOTAL (modificato in TT).

La Figura 1.1 illustra i messaggi visualizzati dai Programmi 1.1 e 1.2 in base ai dati di una bolletta tipo: i due programmi generano il medesimo risultato. Se si memorizza la versione GW-BASIC in formato ASCII è possibile richiamarla, compilarla e mandarla in esecuzione con QuickBASIC senza modifiche, mentre non è possibile compilare od eseguire con QuickBASIC la versione Applesoft, riportata per chi ha una esperienza di Applesoft ed allo scopo di agevolare per coloro che se si sono rivolti a IBM PC od a PC compatibili la conversione da Applesoft a GW-BASIC o BASICA, che sono accettati dal compilatore QuickBASIC.

```
TOTALE   KWH? 293
TOTALE   MC? 53

TOTALE ELETTRICITA = L. 20.66
TOTALE GAS         = L. 26.76
TASSA ENERGIA      = L.  0.06

BOLLETTA TOTALE    = L. 47.48
```

Figura 1.1 Output del Programma 1.1

1.3.2 Calcolo del valore dei titoli

I lettori che seguono il mercato azionario potranno trovare un aiuto nel seguente programma dimostrativo, che potrà essere utilizzato per calcolare il valore delle azioni possedute. Il breve programma è costituito da tre soli blocchi: identificazione del programma, immissione dati e calcolo; l'abbozzo del programma con sole istruzioni REM è il seguente:

```
1   REM ** VALORE DEI TITOLI **
100 REM ** IMMISSIONE DATI E CALCOLO DEL VALORE **
200 REM ** ARROTONDAMENTO E STAMPA DEL VALORE **
```

Il Programma 1.3 è la versione GW-BASIC ed il Programma 1.4 è la versione Applesoft di VALORE DEI TITOLI; in ciascuna la sezione IMMISSIONE DATI E CALCOLO DEL VALORE chiede il numero delle azioni ed il valore di ognuna per ogni titolo e, non appena vengono inseriti i dati, calcola il valore totale delle azioni.

Una volta immessi questi valori per ogni titolo si digiti **0** alla richiesta del numero di azioni e **0**, od un altro numero per il prezzo di ogni azione; il programma procederà verso il blocco ARROTONDAMENTO E STAMPA DEL VALORE, dove il valore del portafoglio verrà arrotondato al centesimo più vicino e stampato.

```
1  REM ** VALORE DEI TITOLI **
2  ' Capitolo 1
3  ' GW-BASIC  File: UQB0103.BAS
9  '
100 REM ** IMMISSIONE DATI E CALCOLO DEL VALORE **
110 CLS: WORTH# = 0: SHARES = .0001
120 ST$ = "##.###.###"
130 WHILE SHARES <> 0
140   INPUT "NUMERO DI AZIONI  "; SHARES
150   INPUT "PREZZO PER AZIONE "; PPS
160   WORTH# = WORTH# + SHARES * PPS
170   PRINT
180 WEND
199 '
200 REM ** ARROTONDA E STAMPA IL VALORE **
210 WORTH# = INT(WORTH# * 100 + .5) / 100
220 PRINT "VALORE TOTALE DEI TITOLI = ";: PRINT USING ST$; WORTH#
230 END
```

Programma 1.3 VALORE DEI TITOLI - GW-BASIC

Entrambe le versioni del programma portano agli stessi risultati. Si è modificata la variabile WORTH della versione GW-BASIC in TWR (Total WoRth) per la versione Applesoft, poiché all'interno della variabile WORTH compare la parola riservata OR ed Applesoft interpreterebbe WORTH come "W OR TH"; non è dunque possibile utilizzare WORTH come variabile Applesoft. Inoltre, Applesoft utilizza HOME anziché CLS del GW-BASIC per pulire lo schermo. Mentre la versione GW-BASIC (Programma 1.3) fa uso di un ciclo WHILE...WEND per ricevere i dati e per calcolare il valore dei titoli, quella Applesoft, non possedendo una struttura WHILE...WEND, deve fondarsi solamente sull'istruzione GOTO alla linea 160 per ricevere i dati e sull'istruzione IF...THEN alla linea 150 per abbandonare il blocco di immissione dati.

Nella versione GW-BASIC si fa uso del simbolo di numero (#) nella variabile WORTH# per dichiararla variabile in doppia precisione, nel caso si abbia un portafoglio di titoli di grande valore. Ancora una volta le differenze nelle istruzioni PRINT ed INPUT determinano l'appropriata spaziatura. Una tipica esecuzione di questi programmi è illustrata in Figura 1.2.

```
1  REM ** VALORE DEI TITOLI **
2  ' Capitolo 1
```

```
3  ' Basic Applesoft  File: UQB0104.BAS
9  :
100 REM ** IMMISSIONE DATI **
110 HOME: TWR = 0
120 INPUT "NUMERO DI AZIONI  ? "; SHARES
130 INPUT "PREZZO PER AZIONE ? "; PPS
140 TWR = TWR + SHARES * PPS
150 IF SHARES = 0 THEN 210
160 PRINT : GOTO 120
199 :
200 REM ** CALCOLO **
210 TWR = INT(TWR * 100 + .5) / 100
220 PRINT "VALORE TOTALE DEI TITOLI = "; TWR
230 END
```

Programma 1.4 VALORE DEI TITOLI - Applesoft

```
NUMERO DI AZIONI  ? 58
PREZZO PER AZIONE ? 62.125

NUMERO DI AZIONI  ? 23
PREZZO PER AZIONE ? 10.250

NUMERO DI AZIONI  ? 44
PREZZO PER AZIONE ? 160

NUMERO DI AZIONI  ? 0
PREZZO PER AZIONE ? 0

VALORE TOTALE DEI TITOLI =        3.841.270
```

Figura 1.2 Output dei Programmi 1.3 ed 1.4

1.4 Tecniche di programmazione

Apprendere le tecniche di programmazione offre l'opportunità di accrescere le proprie potenzialità nella soluzione di problemi, non solo di tipo matematico, ma di qualsiasi tipo.

Il calcolatore è un prezioso strumento per apprendere come ragionare in modo logico e di conseguenza il processo di soluzione dei problemi diviene:

- definizione del problema;
- suddivisione del problema in una serie di sottoproblemi più semplici;
- stesura del programma sotto forma di brevi blocchi funzionali che risolvano i sottoproblemi.

Se viene osservata questa procedura si perviene alla soluzione nel modo più logico, più veloce e più attendibile; nel corso della procedura si diviene più intimamente coscienti delle capacità e limitazioni del calcolatore. Nella risoluzione dei problemi il fattore umano è più importante degli strumenti che si hanno a disposizione.

Il calcolatore è una macchina utilizzata da uomini e quindi questi dovrebbero scrivere programmi leggibili e comprensibili da altri uomini; in questa sezione si proporranno alcune convenzioni utilizzate nel testo per promuovere una buon stile di programmazione ed una buona struttura. Queste convenzioni porteranno gradualmente allo stile di programmazione accettato da QuickBASIC 4.5 (struttura e buon stile sono caratteristiche che dovrebbero essere sempre utilizzate durante la programmazione) e procureranno la standardizzazione e l'ambiente didattico che permetteranno la transizione dalla versione di BASIC familiare a QuickBASIC 4.5 della Microsoft.

1.4.1 Convenzioni e stili

Un calcolatore riceve informazioni, le elabora secondo specifiche istruzioni e fornisce i risultati dell'elaborazione sotto forma di nuove informazioni. Il motivo principale per apprendere la programmazione è la possibilità di utilizzare il calcolatore nel modo più efficiente possibile, infatti i programmi in commercio, pur essendo disponibili per molti scopi, raramente sono scritti in modo da soddisfare tutte le esigenze del singolo individuo: sono piuttosto progettati per essere utilizzati da un pubblico che sia il più vasto possibile allo scopo di incrementare le vendite. Acquisendo la capacità di scrivere i propri programmi, è possibile ottenere dal calcolatore ciò che si vuole e nel modo desiderato.

Durante la programmazione si potrà rapidamente comprendere come definire un problema e suddividerlo in sottoproblemi più semplici e maneggevoli e scrivere una serie di istruzioni che li risolvano; ogni blocco di istruzioni risolve una parte del problema originale, che verrà risolto concatenando i blocchi in un unico programma completo.

È importante che le istruzioni date al calcolatore (il programma) siano scritte in modo da essere ad esso comprensibili ed anche di facile lettura, comprensione ed utilizzo da parte dell'autore e di terzi.

Si sono tentati di scrivere programmi rivolti solamente a sé stessi, senza preoccuparsi della loro logica o struttura, pensando di utilizzarli una sola volta e supponendo che nessun altro

ne farà uso, mentre non è invece da escludere che si possa ottenere un buon programma e lo si desideri utilizzare altre volte, condividerlo con amici o, addirittura, metterlo a disposizione di un intero gruppo di utenti o di una rivista per la pubblicazione.

Il buon stile di programmazione è una buona abitudine che porta ad ottenere buoni programmi.

Questo libro mette a disposizione esempi accuratamente selezionati, che fanno uso di un insieme di convenzioni uniforme e di uno stile strutturato.

QuickBASIC fornisce un'ampia gamma di strutture di programmazione ed agevola anche uno stile di programmazione uniforme quando la sua funzione di controllo di sintassi è in funzione.

1.4.1.1 Istruzioni REM

Ogni programma di questo libro ha inizio con una istruzione REMARK, la prima delle quali riferisce il nome del programma. GW-BASIC e QuickBASIC ammettono anche l'apostrofo (') per indicare l'istruzione REM, forma abbreviata che è comoda quando vengono utilizzate più istruzioni REM nella parte iniziale di un blocco del programma.

Ad esempio: la seconda istruzione REM del blocco di apertura identifica il titolo del libro ed il capitolo, la terza comunica il linguaggio utilizzato ed il nome con il quale il file è stato memorizzato. Altre informazioni descrittive possono essere incluse nelle istruzioni REM del primo blocco del programma: si inserisca dunque qualsiasi informazione si voglia.

Queste istruzioni non vengono eseguite, ma forniscono informazioni descrittive per agevolare l'uso del programma.

Le istruzioni REM non vengono incluse nel programma compilato da QuickBASIC, ma sono relegati al solo codice sorgente visibile sullo schermo. Ad esempio:

```
200 REM ** CICLO PRINCIPALE **
210 WHILE 1
299   '
300 REM ** INIZIALIZZAZIONE DI RIS1 e RIS2 ED IMMISSIONE PAROLA **
```

Ogni blocco funzionale ha inizio con almeno un'istruzione REM indicante in modo chiaro la funzione di quella parte di programma; gli asterischi utilizzati in queste istruzioni hanno lo scopo di agevolare la lettura veloce del programma e l'identificazione rapida delle istruzioni REM che descrivono le funzioni dei blocchi.

In GW-BASIC e QuickBASIC viene utilizzata anche l'apostrofo per inserire commenti al termine di una istruzione eseguibile, come indicato qui di seguito:

```
DEFDBL W     'le variabili in doppia precisione iniziano con W
```

Applesoft non riconosce l'apostrofo quale alternativa a REM; nei programmi Applesoft si utilizzerà un due punti per aggiungere commenti al seguito di una istruzione eseguibile.

1.4.1.2 Parole chiave BASIC

Le parole chiave BASIC (PRINT, INPUT ed altre) in Applesoft devono essere digitate in maiuscolo, mentre in GW-BASIC e BASICA possono essere digitate sia in maiuscolo che in minuscolo; quando viene listato il programma in GW-BASIC o BASICA, le parole chiave appaiono in maiuscolo indipendentemente da come siano state inserite. Anche in QuickBASIC le parole chiave possono essere digitate in maiuscolo o minuscolo e l'editor di QuickBASIC trasformerà automaticamente le minuscole in maiuscole quando la linea che contiene parole chiave in minuscolo viene inserita o quando il cursore viene spostato dalla stessa. In questo libro si utilizzeranno sempre le maiuscole per le parole chiave.

1.4.1.3 Variabili

GW-BASIC, BASICA e QuickBASIC permettono e riconoscono variabili distinte e lunghe fino a 40 caratteri, mentre Applesoft accetta solamente variabili di 8 caratteri, dei quali riconosce solamente i primi due; se si è utenti Applesoft, è dunque necessario porre attenzione a che i primi due caratteri di variabili diverse non coincidano, se si desidera che le stesse vengano riconosciute come distinte una dall'altra. Le variabili in Applesoft devono essere digitate in maiuscolo, in GW-BASIC o BASICA è indifferente, ma, quando listate, appariranno in maiuscolo; in QuickBASIC è possibile digitarle in maiuscolo o in minuscolo e, quando viene listato il programma, appariranno esattamente come sono state inserite.

I programmi Applesoft e GW-BASIC presenti in questo libro utilizzeranno unicamente variabili in maiuscolo; benché ciò non sia necessario in GW-BASIC, i programmi listati mostrano comunque le variabili in maiuscolo, indipendentemente da come siano state inserite in origine. Le variabili QuickBASIC compariranno in minuscolo o in un misto di minuscolo e maiuscolo, come nel listato seguente. Questa convenzione rende i programmi di più facile lettura.

```
FOR numero =  TO LEN(Parola$)
  Lettera$ = MID$(Parola$, numero, 1)
  If Lettera$ >= "A" AND Lettera$ <= "Z" THEN
    .
    .
    .
```

1.4.1.4 Numerazione delle linee

In Applesoft e GW-BASIC sono necessari i numeri di linea: nel blocco di apertura si attribuiscono sempre valori minori di 100, gli altri blocchi iniziano sempre con numeri multipli di 100 ed ogni subroutine utilizzata ha inizio con un numero di linea multiplo di 1000.

In QuickBASIC i numeri di linea non sono necessari e non saranno quindi utilizzati nei programmi QuickBASIC di questo libro.

1.4.1.5 Convenzioni di punteggiatura

Una virgola, un due punti od un punto e virgola sono normalmente seguiti da uno spazio, ad eccezione del caso in cui la punteggiatura sia utilizzata in una stringa e, in programmi non in QuickBASIC, nel caso in cui due punti segua immediatamente un punto e virgola. Queste convenzioni sono illustrate nel seguente listato:

```
210 CLS: WORTH# = 0: SHARES = .0001
430 PRINT: PRINT "TOTALE ELETTRICITA = " "; TELEC
220 PRINT "VALORE TOTALE = ";: PRINT USING ST$; WORTH#
```

QuickBASIC provvede in parte ad una spaziatura automatica in modo da assicurare un aspetto uniforme al programma.

1.4.1.6 Simboli di operazione e di relazione

Un'abbondante spaziatura nei programmi li rende molto più leggibili: in questo libro si pone uno spazio prima e dopo ogni simbolo di operazione (+-*/) o di relazione (= < > <= >= <>), come è illustrato nei seguenti esempi:

```
130 WHILE SHARES <> 0
160   WORTH# = WORTH# + SHARES * PPS
210 WORTH# = INT(WORTH# * 100 - .5) / 100
```

Esiste un'eccezione: quando il segno meno denota un numero negativo non viene seguito da uno spazio.

```
240 FOR HERE = BOTTOM TO TOP STEP -1
```

1.4.1.7 Indentazione dei cicli

Un ciclo FOR...NEXT ha inizio con un'istruzione FOR e termina con un'istruzione NEXT e le linee contenute tra le due vengono chiamate corpo del ciclo. Il formato di FOR intercala uno spazio tra ogni elemento dell'istruzione e, benché in GW-BASIC e QuickBASIC sia opzionale, ogni istruzione NEXT verrà seguita dal nome della variabile utilizzata nell'istruzione FOR, ciò è fatto in questo libro allo scopo di definire in modo chiaro il termine del ciclo.

La indentatura è utilizzata per delineare il corpo di ciascun ciclo: questo è un tipico ciclo FOR...NEXT:

```
410 FOR NUM = 1 TO LEN(PAROLA$)
420 LETTERA$ = MID$(PAROLA$, NUM, 1)
430 IF LETTERA$ < "A" AND LETTERA$ > "Z" THEN 470
440 LSCORE = ASC(LETTERA$) - 64
450 RIS1 = RIS1 + LVALORE
460 RIS2 = RIS2 + LVALORE
470 NEXT NUM
```

GW-BASIC e QuickBASIC permettono anche un ciclo WHILE...WEND, che viene aperto dall'istruzione WHILE e chiuso dalla WEND. Nell'istruzione WHILE è inclusa una condizione ed il corpo del ciclo viene eseguito in continuazione sino a che questa è vera, mentre quando diviene falsa il programma prosegue dalla istruzione successiva alla WEND. Seguono aperture e chiusure di tipici cicli WHILE...WEND.

```
130 WHILE SHARES <> 0
140   INPUT "NUMERO DI AZIONI  ": SHARES
150   INPUT "PREZZO PER AZIONE "; PPS
160   WORTH# = WORTH# + SHARES * PPS
170   PRINT
180 WEND
```

La condizione nell'istruzione while (WHILE SHARES <> 0) è originariamente vera poiché impostata nell'istruzione precedente il ciclo e, sino a che non si inserisce un valore nullo per la variabile SHARES, la condizione sarà soddisfatta ed il ciclo ripetuto.

Per terminare il ciclo si inserisca un valore 0 come numero di azioni, la condizione nell'istruzione WHILE sarà allora divenuta falsa ed il programma uscirà dal ciclo.

1.4.1.8 Istruzioni multiple

Le istruzioni multiple sono utilizzate solamente quando eseguono una funzione composita, quando cioè le istruzioni assolvono insieme un unico compito; come ad esempio quando viene inizialmente pulito lo schermo ed eliminata la linea dei tasti funzione:

```
120 CLS: KEY OFF
```

Nei programmi GW-BASIC uno spazio segue generalmente il due punti di separazione; a volte QuickBASIC provvede ad una riformattazione che pone uno spazio prima e dopo il due punti.

1.4.2 Convenzioni utilizzate

Il Programma 1.5, NUMEROLOGIA, illustra lo stile ed alcune delle convenzioni utilizzate nei programmi GW-BASIC in questo libro: le istruzioni REM sono utilizzate per descrivere il programma e suddividerlo in blocchi funzionali, la parte principale consiste in un ciclo

WHILE...WEND che viene eseguito in continuazione, un ciclo FOR...NEXT è nidificato all'interno di quello WHILE...WEND (si osservi come i corpi dei cicli siano chiaramente definiti dall'indentatura).

Inoltre una subroutine che trasforma in maiuscole la parola inserita è chiamata dalla parte principale del programma ed il valore della parola inserita viene calcolato dopo il ritorno dalla subroutine.

```
REM ** NUMEROLOGIA **
2 ' Capitolo 1
3 ' Microsoft GW-BASIC  File: UQB0105.BAS
9 '
100 REM ** INIZIALIZZA **
110 DEFDBL W            'Le variabili in doppia precisione iniziano in W
120 CLS: KEY OFF
199 '
200 REM ** CICLO PRINCIPALE **
210 WHILE 1
299   '
300   REM ** INZIALIZZAZIONE DI RIS1 E RIS2 E IMMISSIONE PAROLA **
310   RIS1 = 0: RIS2 = 1
320   INPUT "Parola"; PAROLA$
330   IF PAROLA$ = "" THEN END
340   GOSUB 1010
399   '
400   REM ** CALCOLA RIS1 & RIS2 **
410   FOR NUM = 1 TO LEN(PAROLA$)
420     LETTERA$ = MID$(PAROLA$, NUM, 1)
430     IF LETTERA$ < "A" AND LETTERA$ > "Z" THEN 470
440     LVALORE = ASC(LETTERA$) - 64
450     RIS1 = RIS1 + LVALORE
460     RIS2 = RIS2 * LVALORE
470   NEXT NUM
499   '
500   REM ** STAMPA LVALORE E RIPETE **
510   PRINT "LVALORE 1 "; RIS1
520   PRINT "LVALORE 2 "; RIS2
530   PRINT
540 WEND
599 '
1000 REM ** SUBROUTINE: MAIUSCOLE **
1010 ' RENDE PAROLA MAIUSCOLA
1020 FOR CPOS = 1 TO LEN(PAROLA$)
1030   CODE = ASC(MID$(MAROLA$, CPOS, 1))
1040   IF CD >96 AND CD < 123 THEN MID$(PAROLA$, CPOS) = CHR$(CD - 32)
1050 NEXT CPOS
1060 RETURN
```

Programma 1.5 NUMEROLOGIA - GW-BASIC

I caratteri incontrati non appartenenti all'alfabeto non vengono contati nel valore della parola, infatti le linee di programma che calcolano il totale corrente del valore per la parola vengono evitate quando la IF dell'istruzione IF...THEN seguente viene eseguita:

```
430     IF LETTERA$ < "A" AND LETTERA$ > "Z" THEN 470
440     LVALORE = ASC(LETTERA$) - 64
450     RIS1 = RIS1 + LVALORE
460     RIS2 = RIS2 * LVALORE
470 NEXT NUM
```

Al fine di ottenere il valore effettivo di una determinata lettera viene sottratto 64 al codice ASCII della stessa:

```
440     LVALORE = ASC(LETTERA$) - 64
```

Alcuni esempi di calcolo di LVALORE sono:

Lettera B (valore ASCII,66):LVALORE = 66 - 64 = 2

Lettera A (valore ASCII,65):LVALORE = 65 - 64 = 1

Lettera S (valore ASCII,83):LVALORE = 83 - 64 = 19

Lettera I (valore ASCII,73):LVALORE = 73 - 64 = 9

Lettera C (valore ASCII,67):LVALORE = 67 - 64 = 3

Il programma fa uso di due metodi distinti per determinare il valore della parola inserita: quello di RISULTATO 1 è calcolato sommando i valori di LVALORE relativi ad ogni singola lettera, mentre quello di RISULTATO 2 moltiplicandoli. Dall'esempio precedente:

Per la parola BASIC

RISULTATO 1 = 2 + 1 + 19 + 9 + 3 = 34

*RISULTATO 2 = 2 * 1 * 19 * 9 * 3 = 1026*

Il programma permette l'introduzione di parole in lettere maiuscole (A, B, C e così via) od in minuscole (a, b, c e così via), oppure miste in lettere maiuscole e minuscole. Si possono includere spazi, lineette, apostrofi od altri caratteri non alfabetici, ma il programma li ignorerà nel calcolare i valori della parola. Se si utilizza una virgola in una parola è necessario porre la parola tra virgolette; infatti GW-BASIC incontrando una virgola la interpreta come un simbolo di separazione tra più dati in entrata, mentre le virgole comprese tra virgolette non sono trattate come separatori. La subroutine MAIUSCOLE rimpiazza tutte le lettere minuscole con maiuscole, ma lascia tutti gli altri caratteri inalterati.

Benché il Programma 1.5 utilizzi tecniche e convenzioni di programmazione molto simili a quelle già illustrate nei primi programmi, potrebbe non essere familiare l'uso di MID$ come assegnamento alla linea 1040 qui di seguito riportata:

```
IF CD>96 AND CD<123 THEN MID$(WORD$, CPOS)=CHR$(CD - 32)
```

Se il valore di CD è compreso tra 97 e 122 la dichiarazione MID$ successiva al THEN viene eseguita, viene cioè rimpiazzata la lettera minuscola alla posizione CPOS in WORD$

con la corrispondente lettera maiuscola CHR$(CD-32). Si digiti il Programma 1.5 e lo si mandi in esecuzione; la Figura 1.3 mostra i risultati ottenuti.

È possibile modificare il programma in modo che vengano trasformate le maiuscole in minuscole: si potrebbe anche scrivere una subroutine MINUSCOLE per sostituire la MAIUSCOLE e nidificare le linee in cui vengono controllate le lettere e calcolati i valori delle lettere.

```
Parola ? BASIC
RISULTATO 1 34
RISULTATO 2 1026

Parola ? QuickBASIC
RISULTATO 1 95
RISULTATO 2 108785754

Parola ?
```

Figura 1.3 Output del Programma 1.5

1.5 QuickBASIC e GW-BASIC

QuickBASIC accetta generalmente i programmi GW-BASIC che siano stati memorizzati in formato ASCII (con l'opzione A), mentre quando non viene fatto uso di questa opzione la memorizzazione avviene in un formato compresso speciale che QuickBASIC non è in grado di leggere.

Desiderando utilizzare un programma GW-BASIC memorizzato in formato compresso, lo si richiami in GW-BASIC e lo si memorizzi nuovamente con la opzione ,A, come qui illustrato:

```
LOAD"UQB0105"
Ok
SAVE"UQB0105",A
```

Sono evidenti alcune differenze nell'aspetto dei due programmi: la versione QuickBASIC utilizza minuscole per alcune variabili, mentre altre sono miste maiuscole e minuscole, QuickBASIC ha una funzione interna (UCASE$) che trasforma la espressione sotto forma di stringa in sole maiuscole, quindi l'istruzione GOSUB e l'intera subroutine UPCASE del programma GW-BASIC sono rimpiazzate dall'unica istruzione:

```
Parola$ = UCASE$(Parola$)
```

È possibile avere utilizzato in precedenza istruzioni e funzioni che non sono accettate da QuickBASIC ed inoltre alcune istruzioni e funzioni richiedono in QuickBASIC più informazioni non necessarie in GW-BASIC; di seguito vengono elencate le istruzioni e funzioni GW-BASIC non utilizzabili in QuickBASIC. Si esaminino attentamente i contenuti dell'elenco, dato che queste funzioni sono svolte in altro modo o sono inutili in QuickBASIC.

1.5.1 Comandi e funzioni proibite

AUTO Genera automaticamente un numero di linea quando si preme il tasto **Invio**. In QuickBASIC non sono necessari i numeri di linea.

CONT Riprende l'esecuzione, precedentemente interrotta, di un programma. È presente nel menu Esegui di QuickBASIC.

DEF USR Definisce il numero e l'indirizzo di una subroutine chiamata dalla funzione USR. In QuickBASIC le funzioni definite dall'utente sono create ed utilizzate diversamente.

DELETE Elimina linee o gruppi di queste in un programma. In QuickBASIC questa operazione è portata a termine diversamente.

EDIT Predispone al modo edit (redazione) di GW-BASIC: l'editor di GW-BASIC è automaticamente in funzione quando si accede alla finestra di visualizzazione. Nel menu Edit sono disponibili funzioni utili alla redazione.

LIST Elenca sullo schermo il programma in memoria. QuickBASIC mostra il programma corrente nella finestra di visualizzazione.

LLIST Manda alla stampante il programma in memoria. QuickBASIC utilizza il comando Stampa del menu File.

LOAD Richiama in memoria da disco un programma GW-BASIC. QuickBASIC utilizza un comando OPEN (apri) del menu File.

MERGE Richiama in memoria da disco un programma GW-BASIC e lo inserisce nel programma corrente. QuickBASIC utilizza un comando Merge del menu File.

MOTOR Avvia o disattiva il mangianastri. QuickBASIC non accetta comandi per l'utilizzo di mangianastri.

NEW Elimina il programma correntemente in memoria. In QuickBASIC si ottiene il medesimo risultato con il menu File.

RENUM Rinumera le linee del programma GW-BASIC correntemente in memoria. I numeri di linea non sono necessari in QuickBASIC.

SAVE Memorizza un programma su disco. Per questo scopo si utilizzano i comandi Salva e Salva con nome del menu File di QuickBASIC.

USR Chiama una subroutine utente in linguaggio assembler. In QuickBASIC viene utilizzato il comando CALL.

1.5.2 Comandi che richiedono una modifica

Vengono qui di seguito elencati comandi e le istruzioni che devono essere modificati per poter essere utilizzati in QuickBASIC; si consulti il proprio manuale di QuickBASIC per avere informazioni dettagliate sul modo in cui devono essere modificati.

BLOAD/BSAVE Le locazioni di memoria utilizzate da QuickBASIC possono essere differenti.

CALL nome In QuickBASIC l'argomento nome è la procedura SUB che viene chiamata.

CHAIN Le opzioni ALL, MERGE, DELETE e di numero di linea del comando CHAIN di GW-BASIC non sono ammesse in QuickBASIC.

COMMON I comandi COMMON devono essere posti prima di qualunque istruzione eseguibile di QuickBASIC.

DIM Tutte le istruzioni DIM che dichiarano gli array (vettori) statici devono essere poste all'inizio di un programma QuickBASIC.

DRAW In QuickBASIC è necessario che la funzione VAR-PTR$ venga utilizzata con particolari variabili.

PLAY In QuickBASIC è necessario che la funzione VAR-PTR$ venga utilizzata con particolari variabili.

RESUME Se viene incontrato un errore in una funzione su linea unica, QuickBASIC tenta di riprendere l'esecuzione alla stessa linea.

RUN L'oggetto di una istruzione RUN o CHAIN deve essere un file .EXE per i file eseguibili prodotti da QuickBASIC. Mentre nell'ambiente QuickBASIC l'oggetto di RUN o CHAIN è ancora un file .BAS.

L'opzione R di GW-BASIC non è accettata. QuickBASIC non accetta RUN {numero di linea/etchetta di linea}, che riprende l'esecuzione del programma all'etichetta o linea specificata.

1.5.3 Anteprima di QuickBASIC

Il Programma 1.6, NUMEROLOGIA CON GRAFICA, è un'anteprima della programmazione QuickBASIC: si tratta di una modifica del Programma 1.5 (si noti la mancanza dei numeri di linea). È possibile inserire a piacere linee QuickBASIC tra altre già presenti facendo uso dell'editor incorporato nel pacchetto.

```
REM ** NUMEROLOGIA CON GRAFICA **
' Capitolo 1
' Microsoft QuickBASIC 4.5  File: UQB0106.BAS

REM ** INIZIALIZZA **
DEFDBL W: DEFINT A-V      'W: doppia precisione / A-V: interi
CLS

REM ** CICLO PRINCIPALE **
DO
   REM ** INIZIALIZZA RIS1 E RIS2 E RICHIEDE PAROLA **
   ris1 = 0: ris2 = 1
   LOCATE 14, 5: PRINT SPACE$(70)
   LOCATE 14, 5: INPUT "Parola"; Parola$
   IF Parola$ = "" THEN EXIT DO
   Parola$ = UCASE$(Parola$)

   REM ** CALCOLA RIS1 & RIS2 **
   FOR num = 1 TO LEN(Parola$)
     Lettera$ = MID$(Parola$, num, 1)
     IF Lettera$ >= "A" AND Lettera$ <= "Z" THEN
       Lvalore = ASC(Lettera$) - 64
       ris1 = ris1 + Lvalore
       ris2 = ris2 * Lvalore
     END IF
   NEXT num

   REM ** STAMPA I RISULTATI E RIPETE **
   LOCATE 5, 5: COLOR 11, 0: PRINT STRING$(50, 223)
   LOCATE 10, 5: PRINT STRING$(50, 220)
   LOCATE 7, 10: COLOR 7: PRINT SPACE$(60)
   LOCATE 8, 10: PRINT SPACE$(60): LOCATE 7, 10
   PRINT "RISULTATO 1 per "; Parola$; " -> "; : COLOR 25: PRINT ris1
   COLOR 7: LOCATE 8, 10: PRINT "RISULTATO 2 per "; Parola$; " -> ";
   COLOR 28: PRINT ris2: COLOR 7: PRINT
LOOP

END
```

Programma 1.6 NUMEROLOGIA CON GRAFICA

```
RISULTATO 1 per QUICKBASIC ->  95
RISULTATO 2 per QUICKBASIC ->  108785754
Parola?
```

Figura 1.4 Output del Programma 1.6

1.6 Sommario

Questo capitolo ha illustrato in breve l'evoluzione negli anni del BASIC quale linguaggio di programmazione: si sono esaminati i vantaggi e gli svantaggi dei due metodi di traduzione del BASIC nel linguaggio proprio della macchina, con compilatore e con interprete, si sono esaminate le differenze tra il GW-BASIC ed il BASIC Applesoft e si è appreso come GW-BASIC e QuickBASIC siano in molte circostanze più flessibili rispetto ad Applesoft.

Questo capitolo ha anche mostrato le convenzioni e lo stile di programmazione che ci si devono aspettare nel seguito di questo libro; quando le convenzioni sono uniformi i programmi ed i testi risultano più leggibili e comprensibili.

I programmi sono suddivisi in blocchi funzionali, vengono utilizzate le istruzioni REM per documentare i blocchi del programma, compaiono istruzioni multiple in un' unica linea solamente se sono in palese relazione l'una con l'altra, viene fatto uso di nomi descrittivi per le variabili e di una spaziatura uniforme ed appositamente studiata per una più agevole comprensione.

Questo capitolo ha messo il lettore in grado di poter comprendere le istruzioni le funzioni QuickBASIC che verranno utilizzate nel seguito del libro.

Capitolo 2

Introduzione a QuickBASIC 4.5

2.1 Copie di sicurezza
2.2 Prime nozioni
2.3 Realizzazione di un programma QuickBASIC
2.4 Utilizzo di un programma QuickBASIC già memorizzato
2.5 Utilizzo di un programma GW-BASIC già memorizzato
2.6 Altri menu
2.7 Sommario

Questo capitolo illustra come utilizzare il dischetto programmi di QuickBASIC e guida nelle operazioni di stesura, memorizzazione ed esecuzione di tre tipi di programmi:

- programmi QuickBASIC nuovi;
- programmi QuickBASIC precedentemente memorizzati;
- programmi GW-BASIC precedentemente memorizzati in formato ASCII.

Si apprenderà come utilizzare i menu File ed Esegui per scrivere, memorizzare, stampare, compilare ed eseguire il programma; verranno commentati i riquadri di comunicazione che compaiono quando le opzioni scelte nel menu necessitano di maggiori informazioni e verranno introdotte semplici tecniche di redazione e l'uso della guida in linea.

Il compilatore QuickBASIC Microsoft richiede almeno 384K di memoria ed una capacità minima dell'unità a dischetti di 720K, ma Microsoft consiglia, per ottenere una migliore prestazione, una unità a disco fisso e 640K di memoria.

Le possibili configurazioni hardware del calcolatore con le quali può essere eseguito QuickBASIC sono molte e sarebbe difficile e fonte di confusione trattarne dettagliatamente ognuna; in questo capitolo si assume quindi che si disponga di due unità a dischetti e non di una più un disco fisso. Se si possiede un disco fisso le operazioni sono semplificate: si copino semplicemente tutti i file di QuickBASIC sullo stesso e si mandi in esecuzione

QuickBASIC da questo; alcune delle schermate potranno differire lievemente da quelle illustrate in questo libro. In questo capitolo si utilizzano i seguenti comandi MS-DOS:

DIR	elenca i file sul disco;
FORMAT	formatta un disco;
DISKCOPY	copia tutti i file da un disco ad un altro;
COPY	copia dei file selezionati.

Si presume una conoscenza almeno parziale del MS-DOS, in caso contrario prima di procedere si consulti il proprio manuale.

Questo libro tratta esclusivamente la versione 4.5 di QuickBASIC; poichè dispone di menu di facile uso, non sarà necessario imparare a memoria tutti i comandi associati alla maggior parte degli altri compilatori: si potrà redigere, compilare, eseguire, correggere e ricompilare programmi senza abbandonare l'ambiente QuickBASIC. Se sono state utilizzate precedenti versioni di QuickBASIC si troverà la versione 4.5 molto più potente e facile da padroneggiare.

2.1 Copie di sicurezza

Dapprima è bene effettuare in uno dei due modi riportati di seguito una copia di sicurezza dei dischi originali di QuickBASIC, che devono essere quindi riposti in un luogo sicuro nel caso fosse necessario ripetere le copie in un secondo tempo.

- Si esegua il file SETUP.EXE che si trova sul disco originale Setup/Microsoft Express, il quale installa QuickBASIC su disco fisso o su sistema a dischetti; durante l'installazione si selezionino le configurazioni proposte (setup semplificato); nella prima parte di questo libro si utilizzeranno i menu semplificati, ma sarà possibile modificare in seguito le opzioni di installazione da QuickBASIC qualora fosse necessario. Se si utilizza questo metodo di installazione per un sistema a dischetti, saranno necessari cinque dischetti vuoti formattati da 360K oppure tre da 720K; il programma setup fornirà una guida completa delle operazioni necessarie alla copia dei dischetti.
- Si utilizzi il comando MS-DOS DISKCOPY per copiare ogni dischetto originale di QuickBASIC su un dischetto di riserva.

I dischetti originali contengono molti file, ma, per i primi capitoli, sarà necessario il solo dischetto programmi contenente i file QB.EXE e QB45QCK.HLP; il file QB.EXE costituisce l'ambiente di sviluppo QuickBASIC ed il file QB45QCK.HLP contiene le informazioni della guida in linea.

Indipendentemente dal metodo utilizzato per la creazione delle copie di sicurezza dei dischetti originali di QuickBASIC, si dovrebbe ora preparare un dischetto di lavoro, utile

nel corso di questo libro; si formatti dunque con MS-DOS un dischetto nuovo come disco di sistema con il seguente comando:

```
FORMAT A: /s
```

Si copi quindi, con il comando COPY, il contenuto dischetto programmi su quello appena formattato; lo schermo, qualora venisse dato il comando DIR in MS-DOS per il dischetto di lavoro, dovrebbe elencare i seguenti file:

```
COMMAND COM
QB      EXE
QB45QCK HLP
```

Il file COMMAND.COM proviene da MS-DOS, mentre QB.EXE e QB45QCK.HLP provengono dal dischetto programmi; nel seguito ci si riferirà a questo dischetto di lavoro con il nome dischetto di lavoro QB. Con il file COMMAND.COM sul dischetto di lavoro QB, si potrà inizializzare il calcolatore senza dover utilizzare il dischetto di sistema per caricare MS-DOS ogniqualvolta si voglia accedere a QuickBASIC, tuttavia, se si dispone di dischetti da 360K, non si ha spazio a sufficienza per gli altri file oltre a QB.EXE, nel qual caso sarebbe necessario caricare il DOS da un dischetto di sistema prima di utilizzare il dischetto di lavoro ed effettuare una sostituzione di disco per ottenere l'help.

2.2 Prime nozioni

Inizialmente verrà adottato il sistema di menu semplificato (anche se nelle figure saranno riportati indifferentemente), che contiene tutti i comandi utili all'apprendimento di QuickBASIC, sino a che non sia necessario quello completo. In questo capitolo sono descritti tre metodi per eseguire un programma QuickBASIC; le istruzioni presumono che il dischetto di lavoro QB si trovi nell'unità A e che l'unità B sia adibita alla memorizzazione dei file di programmi e dati; tuttavia, se si dispone di unità a dischetti con designazioni diverse, si operino le adeguate sostituzioni alle lettere A e B. Se si dispone di un sistema con due unità a dischetti, i passi per creare ed eseguire un programma QuickBASIC nuovo sono i seguenti:

- si inserisca il dischetto di lavoro QB nell'unità A;
- si inserisca un dischetto formattato vuoto nell'unità B;
- si entri nell'editor di QuickBASIC con il comando QB;
- si digiti il nuovo programma;
- si memorizzi il programma sul dischetto nell'unità B;
- si esegua il programma.

Per eseguire un programma precedentemente memorizzato:

- si inserisca il dischetto di lavoro QB nell'unità A;
- si inserisca il dischetto con il programma QuickBASIC nell'unità B;
- si richiami il programma dalla linea di comando MS-DOS (A) o dal menu File di QuickBASIC;
- si esegua il programma.

Per eseguire un programma GW-BASIC che sia stato memorizzato in formato ASCII:

- si inserisca il dischetto di lavoro QB nell'unità A;
- si inserisca il dischetto contenente il programma GW-BASIC in formato ASCII nell'unità B;
- si richiami dalla linea di comando di MS-DOS o dal menu File di QuickBASIC;
- si rimuova il dischetto contenente il programma GW-BASIC dall'unità B;
- si inserisca nell'unità B il dischetto sul quale si vuole memorizzare la versione QuickBASIC del programma;
- si memorizzi il programma sul dischetto nell'unità B;
- si esegua il programma.

Ognuno di questi metodi è trattato più dettagliatamente nelle sezioni che seguono.

2.3 Realizzazione di un programma QuickBASIC

Con il dischetto di lavoro QB si digiterà, memorizzerà ed eseguirà un programma che simula il lancio di alcune monete; le simulazioni sono potenti strumenti per assolvere molti compiti, alcuni semplici, altri complicati; questo in particolare è semplice in modo da non doversi occupare troppo del suo scopo e per meglio seguire il procedimento di redazione ed esecuzione del programma.

Il Programma 2.1, LANCIO MONETE, è costituito da tre brevi blocchi, il primo dei quali contiene solamente istruzioni REM che descrivono il programma, il secondo chiede il numero di monete che si desidera lanciare ed il blocco finale opera la simulazione del lancio. Ci si assicuri in primo luogo che il dischetto di lavoro QB si trovi nell'unità di default (nell'esempio l'unità A), si inserisca un dischetto formattato nuovo nell'unità B ed al prompt del DOS (A>) si digitino le lettere QB, maiuscole o minuscole:

```
A>QB
```

Questo indica al calcolatore di caricare QB.EXE, il file di QuickBASIC. Si prema il tasto **Invio**: caricato QuickBASIC, compare la schermata illustrata nella Figura 2.1: in un riquadro di comunicazione sovrapposto alla schermata di apertura si trovano un messaggio di benvenuto ed un avviso di Copyright.

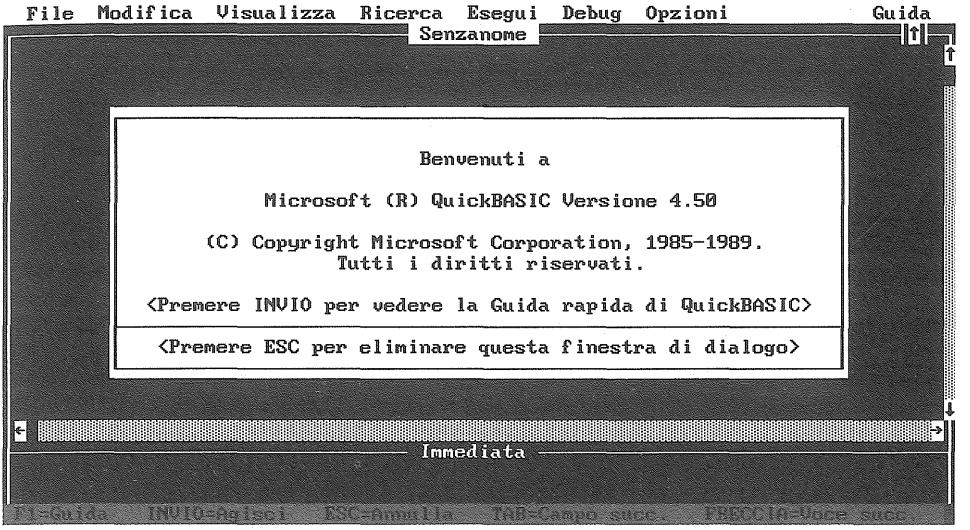


Figura 2.1 Schermata di apertura

I riquadri di comunicazione vengono aperti nell'ambiente QuickBASIC nei momenti opportuni allo scopo di procurare suggerimenti riguardanti le informazioni richieste da QuickBASIC per procedere al passo successivo.

Dalle ultime due linee è possibile comprendere che in questo momento vi sono due scelte possibili:

- premere **Invio**, per vedere la guida rapida di QuickBASIC;
- premere **Esc**, per eliminare il riquadro di comunicazione.

Si prema ora il tasto **Esc** in modo da poter digitare il programma LANCIO MONETE.

```
REM ** LANCIO MONETE **
' Capitolo 2
' Microsoft QuickBASIC  4.5 File: UQB0201.BAS

REM ** Richiede il numero di lanci **
' Chiede il numero di monete da lanciare
CLS : RANDOMIZE TIMER
INPUT "Quante monete devo lanciare"; lanci
PRINT
REM ** Lancia le monete il numero di volte richiesto **
```

```
FOR contatore = 1 TO lanci
   lancio = INT(2 * RND(1))
   IF lancio = 0 THEN PRINT "TESTA", ELSE PRINT "CROCE",
NEXT contatore
END
```

Programma 2.1 LANCIO MONETE

2.3.1 Ambiente schermo di QuickBASIC

Quando viene premuto il tasto **Esc**, viene eliminato il riquadro di comunicazione lasciando spazio alla schermata illustrata nella Figura 2.2 e l'editor di QuickBASIC attende che si digiti il nuovo programma.

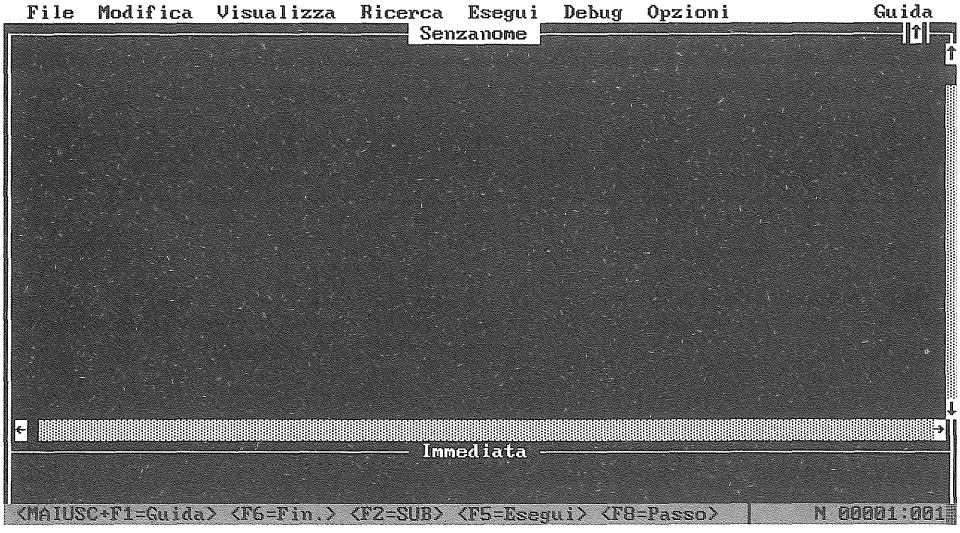


Figura 2.2 Ambiente schermo di QuickBASIC

La grande area rettangolare vuota è detta finestra di visualizzazione ed è la zona in cui verrà digitato il testo; il cursore (il trattino lampeggiante) indica dove apparirà il testo.

Se è stato installato un mouse il suo puntatore si troverà vicino al centro dello schermo.

La finestra di visualizzazione può essere suddivisa in due finestre separate: se la prima contiene il testo del file sorgente (ad esempio la parte principale del programma), la seconda può essere aperta ed utilizzata per mostrare un'altra porzione del programma (ad esempio un sottoprogramma). È possibile redigere il testo in entrambe le finestre, ma poichè il nuovo programma è semplice e breve se ne utilizzi almeno per il momento solamente una.

Il nome del file in corso di redazione è indicato nella parte superiore della finestra di visualizzazione nella linea di intestazione, ma, poichè al nuovo programma non è ancora stato attribuito alcun nome, è indicato "Senzanome" anzichè un nome di file.

L'informazione nella linea di comando è evidenziata, cioè viene visualizzata in modo video inverso.

L'area vuota lunga e rettangolare vicina all'estremo inferiore dello schermo è detta finestra immediata: vi possono essere inseriti ed eseguiti direttamente comandi QuickBASIC.

La barra dei menu nella parte superiore dello schermo contiene i nomi dei menu del sistema semplificato di QuickBASIC: File, Modifica, Visualizza, Ricerca, Esegui, Debug, Opzioni e Guida.

Le barre di scorrimento alle estremità destra ed inferiore della finestra attiva indicano la posizione relativa nel file e possono essere rese attive o meno con il riquadro di comunicazione del menu Opzioni, trattato nel Capitolo 4.

La barra di riferimento che si trova all'estremità inferiore dello schermo e le informazioni in essa contenute sono fornite per agevolare le decisioni da prendere nel corso dell'esecuzione del programma.

2.3.2 Creazione del programma LANCIO MONETE

Si digiti il Programma 2.1 nella finestra di visualizzazione; se vengono commessi alcuni errori, vi sono due metodi facili per rimediare:

- premendo il tasto **Backspace** per eliminare il carattere alla sinistra del cursore;
- premendo il tasto **Canc** per eliminare il carattere sul cursore.

I tasti freccia (su, giù, destra, sinistra) spostano il cursore sopra ai caratteri senza cancellarli; si consiglia comunque, durante la redazione del programma, di provare ad utilizzare tutte queste funzioni.

Una trattazione più completa dell'editor di QuickBASIC e del suo utilizzo è svolta nel Capitolo 4.

Una volta digitato il programma, lo si legga alla ricerca di eventuali errori e, nel caso se ne trovassero, si porti il cursore nella posizione appropriata e li si correggano.

Anzichè eseguire il programma immediatamente, si dovrebbe prima memorizzarlo, altrimenti potrebbe presentarsi qualche errore logico causando malfunzionamenti e si potrebbe essere costretti a resettare il calcolatore cancellando il programma dalla memoria.

Per evitare di dover digitare nuovamente il programma da tastiera, lo si memorizzi prima, in modo da poterlo eventualmente richiamare dal menu File nel modo che verrà descritto nel seguito del capitolo.

2.3.3 Memorizzazione del programma LANCIO MONETE

Una volta sicuri di aver digitato correttamente il programma, si prema la combinazione di tasti **Alt+F** per accedere al menu File, che comparirà in prossimità dell'estremo superiore sinistro dello schermo; si osservi che il comando Nuovo programma è evidenziato e che nella barra di riferimento appare il seguente messaggio:

```
Toglie dalla memoria il programma attualmente caricato
```

Premendo il tasto **Freccia Giù** l'evidenziazione si sposta verso il basso di una linea ed il risultato dell'eventuale selezione di ciascuna voce evidenziata è illustrato nella barra di riferimento.

Si controlli l'intero elenco di comandi premendo il tasto **Freccia Giù**; la Tabella 2.1 elenca le informazioni che si osservano nella barra di riferimento. Premendo il tasto **Freccia Giù** mentre l'evidenziazione si trova sull'ultima voce, l'evidenziazione si sposta all'inizio dell'elenco.

Tabella 2.1 Informazioni nella barra di riferimento per il menu File

Comando del menu	Informazione nella barra di riferimento
Nuovo programma	Toglie dalla memoria il programma attualmente caricato
Apri programma	Carica in memoria un nuovo programma
Salva con nome	Salva il modulo corrente col nome e formato specificati
Stampa	Stampa il testo o modulo specificato
Esci	Esce da QuickBASIC e torna al livello DOS

Si prema il tasto **Freccia Giù** sino a che l'evidenziazione non si trovi sul comando Salva con nome, come illustrato nella Figura 2.3; qualora lo si fosse oltrepassato, si utilizzi il tasto **Freccia Su** per spostare verso l'alto l'evidenziazione.

Premendo il tasto **Freccia Su** quando l'evidenziazione è sulla prima voce, questa si porta al termine dell'elenco.

Il comando Salva con nome è seguito da tre puntini di sospensione (...) come i comandi Apri programma e Stampa: i puntini indicano la necessità di ulteriori informazioni qualora il comando venisse selezionato; se invece al termine di un comando non compaiono puntini, questo viene eseguito immediatamente una volta selezionato senza necessità di alcuna informazione. Nel seguito della trattazione di questo libro saranno riportati i nomi dei comandi senza puntini.

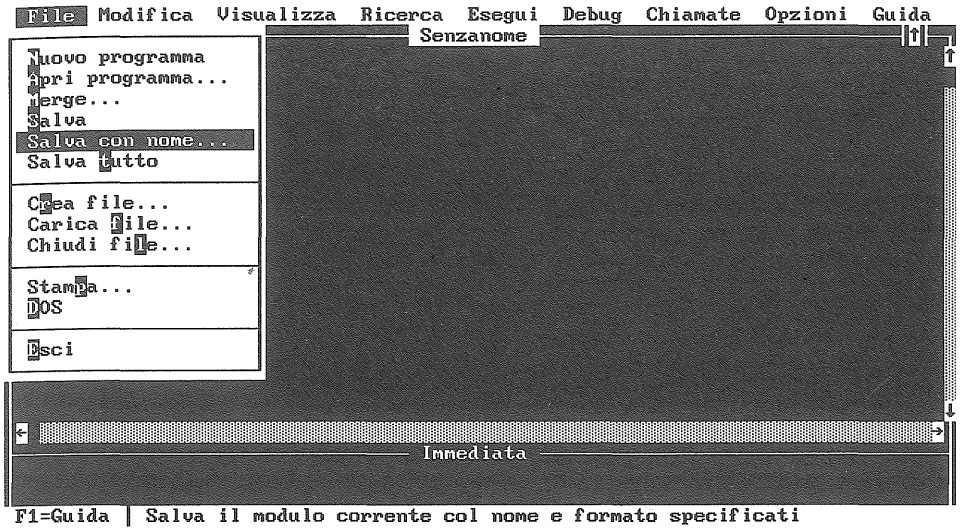


Figura 2.3 Menu File con Salva con nome evidenziato

Si prema il tasto **Invio** mentre è evidenziato il comando Salva con nome: appare un riquadro di comunicazione che richiede le informazioni necessarie per portare a termine la memorizzazione ed il cursore lampeggia alla destra della scritta "Nome del file:", come illustrato nella Figura 2.4.

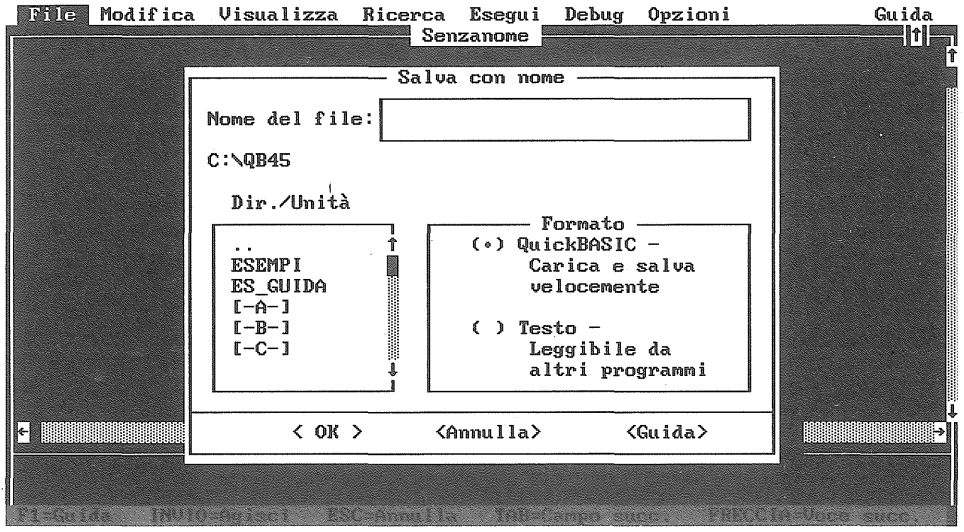


Figura 2.4 Riquadro di comunicazione Salva con nome

Si osservi il percorso di default sotto a "Nome del file:"; se non si digitasse un differente percorso nel nome del file, questo verrebbe in questo caso memorizzato nell'unità A.

Si digiti il nome del file completo; per memorizzare il programma nell'unità B con il nome UQB0201.BAS, si inserisca:

```
B:UQB0201
```

QuickBASIC apporrà automaticamente l'estensione .BAS, ma, se si desiderasse memorizzare il programma con una differente estensione, ad esempio .PRO, si dovrebbe digitare per esteso:

```
B:UQB0201.PRO
```

I nomi di file devono essere conformi al formato MS-DOS: la parte principale non può essere costituita da più di otto caratteri, mentre l'estensione può contenere al più tre caratteri.

Quando si è digitato il nome del file, si prema il tasto **Tab** ed il cursore si sposterà al primo elemento dell'elenco delle directory e delle unità (Dir./Unità).

Premendo nuovamente il tasto **Tab**, il cursore si sposta tra le parentesi della prima voce nell'elenco dei formati; il punto sopra al cursore indica che questa voce è correntemente selezionata.

Premendo una volta il tasto **Freccia Giù**, il cursore si sposterà alla seconda voce, "Testo-Leggibile da altri programmi", che è quella desiderata.

All'estremo inferiore del riquadro di comunicazione sono disposti tre pulsanti: OK, Annulla e Guida. Si osservi che OK è evidenziato, le scelte illustrate nel riquadro di comunicazione sono attive e se si premesse il tasto **Invio** verrebbero utilizzate le seguenti selezioni:

```
Nome del file: B:UQB0201.BAS
Formato        Testo
```

Poiché queste sono le selezioni desiderate, si prema il tasto **Invio** per memorizzare il file. Dopo aver memorizzato il programma si potrebbe volere stampare il listato con la stampante.

Per fare ciò, si acceda al menu File premendo i tasti **Alt+F**, si sposti l'evidenziazione al comando Stampa con il taso **Freccia Giù** e si prema **Invio**; comparirà il riquadro di comunicazione illustrato nella Figura 2.5.

È possibile scegliere una della tre opzioni di stampa per portare a termine le operazioni elencate in Tabella 2.2: poiché non si è selezionato alcun testo, la prima opzione non può essere utilizzata (la selezione del testo verrà trattata nel Capitolo 4), mentre, essendo questo programma breve ed interamente contenuto nella finestra attiva, per stampare il programma potrà essere utilizzata una qualunque delle ultime due opzioni nel riquadro.

Il punto tra le parentesi indica che è attiva l'opzione Modulo corrente. Si prema il tasto **Invio** per stampare il file.

Tabella 2.2 Scelte nel riquadro di comunicazione Stampa

Selezione	Operazione eseguita
Testo selezionato	Stampa solamente il testo selezionato (evidenziato) nella finestra attiva
Finestra attiva	Stampa il contenuto della finestra attiva, in genere un modulo principale completo, una procedura FUNCTION oppure una procedura SUB
Modulo corrente	Stampa il contenuto del file associato alla finestra attiva

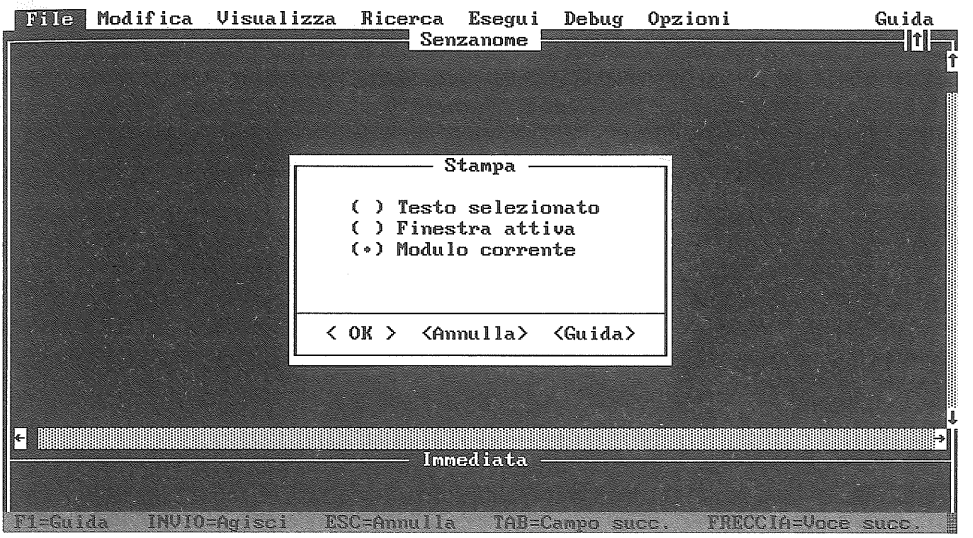


Figura 2.5 Riquadro di comunicazione per la stampa

2.3.4 Esecuzione del programma LANCIO MONETE

I programmi vengono mandati in esecuzione con il menu Esegui; Premendo i tasti **Alt+E**, compare in una finestra il menu Esegui con il comando Avvia evidenziato, come illustrato nella Figura 2.6. È possibile selezionare un comando del menu Esegui premendo i tasti **Freccia Su** o **Freccia Giù**, evidenziando così il comando desiderato; si prema il tasto **Freccia Su** percorrendo tutti i comandi e si osservino le informazioni riportate nella barra di riferimento. Nella Tabella 2.3 sono elencati i comandi e le relative descrizioni. Si evidenzi nuovamente il comando Avvia e si prema **Invio** per eseguire il programma.

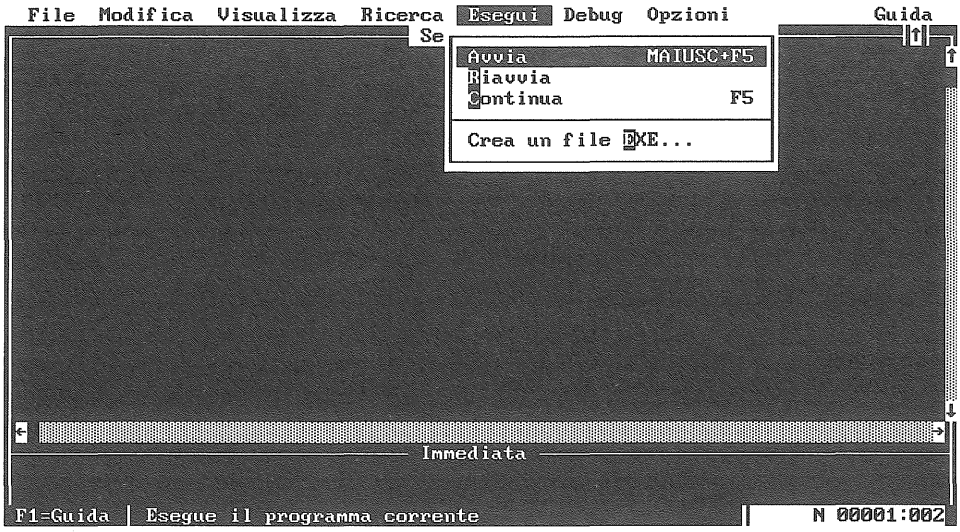


Figura 2.6 Menu Esegui con Avvia evidenziato

Il programma è quindi rapidamente esaminato alla ricerca di errori e, se non ve ne sono, lo schermo viene vuotato ed appare la richiesta mostrata nella Figura 2.7, dove è illustrata un'esecuzione con 50 monete; i risultati, poiché TESTA o CROCE sono selezionati in modo casuale, differiranno. La pulizia dello schermo e l'inizializzazione del generatore di numeri casuali sono portate a termine dalla seguente linea di programma:

```
CLS: RANDOMIZE TIMER
```

Tabella 2.3 Contenuti della barra di riferimento per il menu Esegui

Comando del menu	Informazione nella barra di riferimento
Avvia	Esegue il programma corrente
Riavvia	Azzera le variabili in preparazione al ripristino dell'esecuzione passo a passo
Continua	Continua l'esecuzione dopo una pausa (break)
Crea un file EXE	Crea un file eseguibile su disco

```
Quante monete devo lanciare? _

Quante monete devo lanciare? 50
TESTA     TESTA     CROCE     TESTA     CROCE
TESTA     CROCE     TESTA     TESTA     CROCE
```

```
CROCE     CROCE     CROCE     CROCE     TESTA
TESTA     CROCE     TESTA     TESTA     CROCE
TESTA     TESTA     TESTA     CROCE     CROCE
CROCE     TESTA     CROCE     TESTA     CROCE
TESTA     CROCE     CROCE     CROCE     TESTA
CROCE     TESTA     CROCE     CROCE     CROCE
CROCE     TESTA     TESTA     TESTA     CROCE
CROCE     TESTA     CROCE     CROCE     TESTA
Premere un tasto per continuare
```

Figura 2.7 Output del programma LANCIO MONETE

Quando l'esecuzione sarà completata, apparirà, all'estremità inferiore dello schermo, il messaggio "Premere un tasto per continuare" e, premendo quindi un qualunque tasto, si tornerà alla finestra di visualizzazione con il programma sempre presente.

Che cosa avverrebbe se vi fosse qualche errore? Si pensi al caso in cui fosse stata omessa l'istruzione NEXT nel ciclo FOR...NEXT utilizzato nel programma.

Per introdurre tale errore, si sposti il cursore sulla N dell'istruzione NEXT nel blocco che lancia le monete.

```
            REM ** Lancia le monete il numero di volte richiesto **
            FOR contatore = 1 TO lanci
              lancio = INT(2 * RND(1))
              IF lancio = 0 THEN PRINT "TESTA", ELSE PRINT "CROCE",
cursore>    NEXT contatore
            END
```

Si cancelli la linea premendo la combinazione di tasti **Ctrl+Y**: QuickBASIC individuerà l'errore quando si tenterà l'esecuzione del programma, che ora si presenta così:

```
REM ** Lancia le monete il numero di volte richiesto **

FOR contatore = 1 TO lanci
  lancio = INT(2 * RND(1))
  IF lancio = 0 THEN PRINT "TESTA", ELSE PRINT "CROCE",
END
```

Si acceda nuovamente al comando Avvia del menu Esegui e si prema il tasto **Invio**; questa volta, quando il programma viene esaminato alla ricerca di errori, viene individuato un errore: compare in prossimità del centro dello schermo un riquadro di comunicazione di errore, che descrive l'errore ed evidenzia il pulsante OK, come illustrato nella Figura 2.8 seguente. Si osservi anche che la variabile lanci nell'istruzione FOR è evidenziata, indicando che QuickBASIC non è in grado di determinare dove si trovi il termine del ciclo. Per correggere l'errore, si chiuda innanzitutto il riquadro di comunicazione premendo il tasto

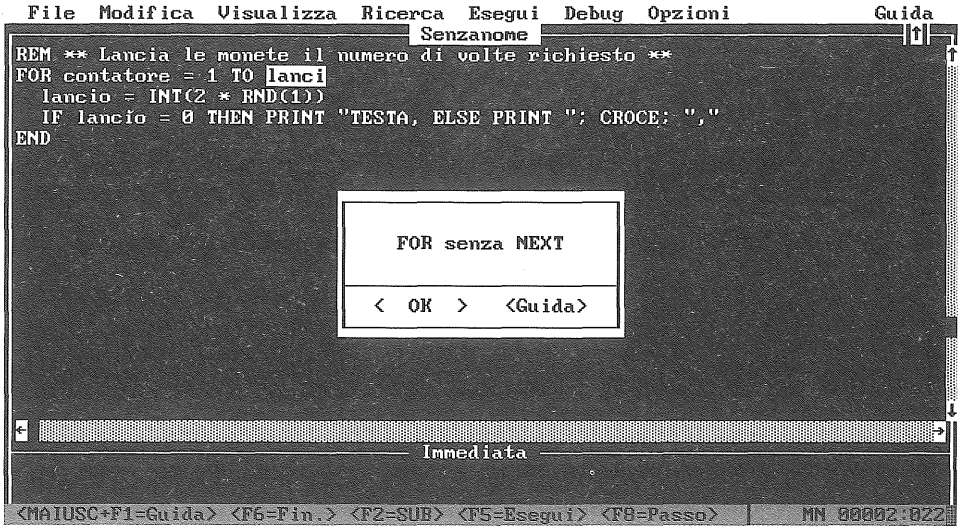


Figura 2.8 Riquadro di comunicazione di errore

Invio con il pulsante OK evidenziato; il riquadro scomparirà ed il cursore apparirà sotto la E dell'istruzione END. Si prema il tasto **Invio** in modo da inserire una linea vuota sopra a quella contenente l'istruzione END, si porti il cursore all'inizio della linea vuota con il tasto **Freccia Sinistra** e si digiti nuovamente l'istruzione END nella sua corretta posizione all'interno del programma.

```
IF lancio = 0 THEN PRINT "TESTA", ELSE PRINT "CROCE",
NEXT contatore

END
```

Dopo aver portato a termine la correzione, si utilizzi il comando Salva con nome del menu File per memorizzare il programma.

2.4 Utilizzo di un programma QuickBASIC già memorizzato

Il metodo per eseguire un programma QuickBASIC già memorizzato (ovviamente disponendo già di un programma QuickBASIC da utilizzare) è il più semplice dei tre: ci si assicuri che il dischetto di lavoro QB si trovi nell'unità A ed il dischetto contenente il Programma 2.1 nell'unità B.

2.4.1 Richiamo di un programma

È possibile richiamare il Programma 2.1 in due modi.

- Richiamando dalla linea di comando del DOS sia QuickBASIC che il programma, includendo nel comando il nome del programma.

  ```
  A>QB B:UQB0201
  ```

  In questo modo si richiama QuickBASIC dall'unità A ed il programma dall'unità B e si accede all'editor con il programma nella finestra di visualizzazione senza dover utilizzare il menu File; è possibile eseguire il programma con il comando Avvia del menu Esegui.

- Richiamando dalla linea di comando del DOS solamente QuickBASIC.

  ```
  A>QB
  ```

Quando QuickBASIC è caricato in memoria, si accede al menu File premendo i tasti **Alt+F**; si porti l'evidenziazione sul comando Apri programma e si prema **Invio**; comparirà il riquadro relativo (Figura 2.9). Si acceda al file digitando B:UQB0201 nella finestra "Nome del file:" nella parte superiore dello schermo e premendo **Invio**. Con il secondo metodo, sarebbe possibile accedere al file anche nel modo seguente:

- si prema il tasto **Tab** spostando il cursore al primo elemento della casella "Dir./Unità";
- si prema il tasto **Freccia Giù** due volte in modo da evidenziare la seconda voce ([-B-]) della casella "Dir./Unità": l'informazione nella casella "Nome del file:" cambia in

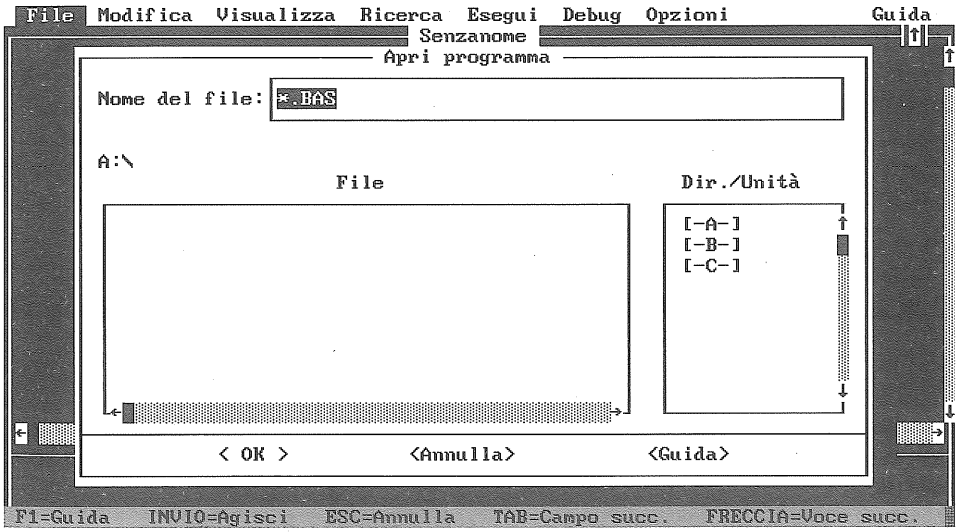


Figura 2.9 Riquadro di comunicazione Apri programma

modo da mostrare il carattere jolly per la selezione dei file .BAS sul dischetto nell'unità B:*.BAS, come illustrato nella Figura 2.10;

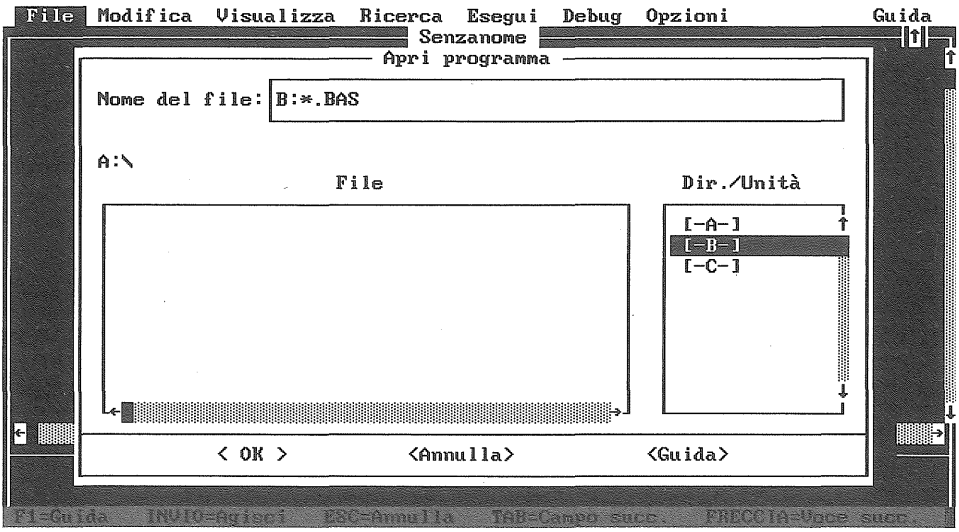


Figura 2.10 Riquadro di comunicazione Apri programma con l'unità B selezionata

- si prema il tasto **Invio** per vedere l'elenco dei file .BAS memorizzati sul dischetto nell'unità B: comparirà, come nella Figura 2.11, un solo file;
- si prema il tasto **Tab** per spostare il cursore alla lista dei file;
- si prema il tasto **Freccia Giù** fino ad evidenziare UQB0201.BAS e quindi **Invio** per caricare il programma.

Indipendentemente dal metodo utilizzato, quando si preme il tasto **Invio** dopo aver digitato (oppure selezionato) il nome del file, il programma verrà caricato in memoria e quindi visualizzato.

2.4.2 Esecuzione di un programma

Si acceda, mediante i tasti **Alt+E**, al menu Esegui, che comparirà in una finestra nella quale è evidenziato il comando Avvia, come illustrato in Figura 2.6; a questo punto si mandi il programma in esecuzione premendo il tasto **Invio** mentre il comando Avvia è ancora evidenziato.

Per accertarsi della effettiva casualità dei risultati, si consiglia di eseguire il programma più volte.

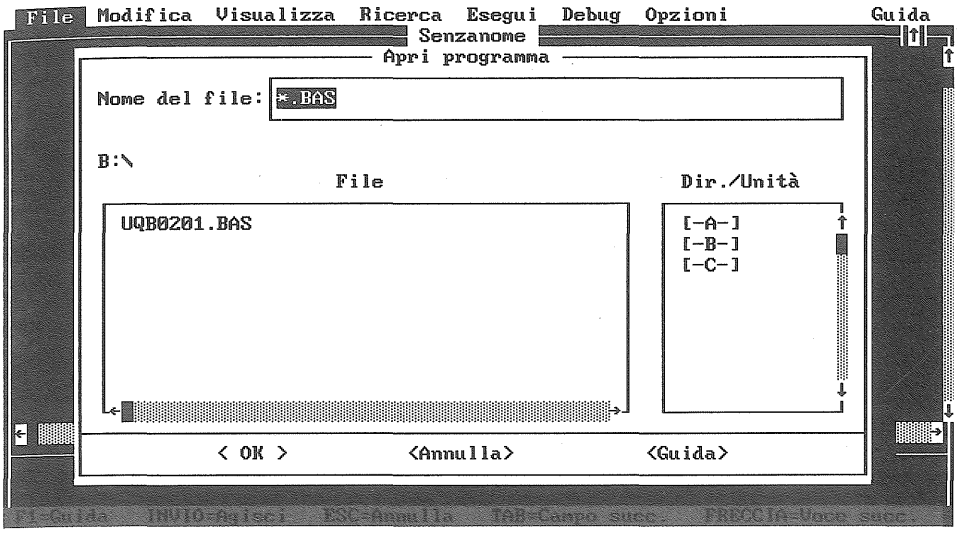


Figura 2.11 Riquadro di comunicazione Apri programma con l'elenco dei file

2.5 Utilizzo di un programma GW-BASIC già memorizzato

Si prosegua la lettura solo se si utilizza GW-BASIC. Affinché QuickBASIC possa compilare ed eseguire un programma GW-BASIC, questo deve essere in formato ASCII; se non si è dunque già memorizzato il Programma 1.3, VALORE DEI TITOLI, in formato ASCII, ci si fermi ora, si ritorni a GW-BASIC, si richiami o digiti il Programma 1.3 e lo si memorizzi in formato ASCII con il seguente comando:

```
SAVE "B:UQB01103.BAS",A
```

Tenendo il dischetto contenente il programma nell'unità B ed il dischetto di lavoro QB nell'unità A, si digiti **QB** al prompt del DOS e si prema **Invio** e quindi **Esc** per visualizzare l'ambiente schermo di QuickBASIC illustrato nella Figura 2.2.

2.5.1 Richiamo di un programma GW-BASIC

Si premano i tasti **Alt+F** per accedere al menu File, quindi il tasto **Freccia Giù** sino ad evidenziare il comando Apri programma, come illustrato nella Figura 2.12. Comparirà il riquadro di comunicazione Apri programma (si veda Figura 2.9); si digiti ora nella casella

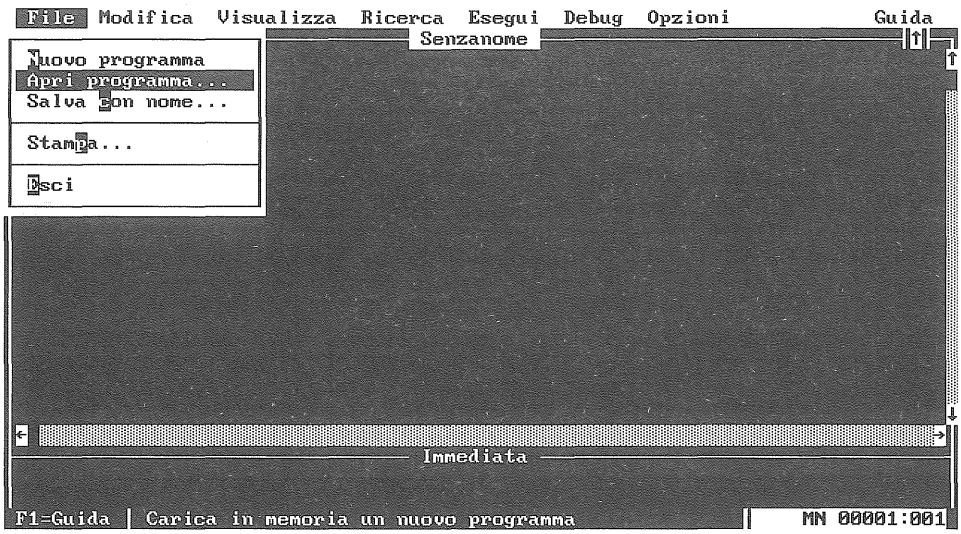


Figura 2.12 Menu File con il comando Apri programma evidenziato

"Nome del file:" ed il percorso completo del programma (comprendente l'unità contenente il file UQB0103 ed il nome del file).

```
Nome del file:   B:UQB0103
```

Si prema quindi **Invio** per richiamare il programma, che la Figura 2.13 illustra già caricato con numeri di linea e tutto il resto. QuickBASIC ha operato due lievi modifiche: ha inserito uno spazio tra CLS ed il due punti alla linea 110 e tra il punto e virgola ed il due punti alla linea 220. È possibile spostare il cursore all'interno del programma per modificarlo: premendo il tasto **Freccia Giù** si sposta verso il basso di una linea, mentre, se il cursore si trova sull'ultima linea visualizzata, così facendo si fa scorrere il programma di una linea verso l'alto (o da un altro punto di vista la finestra di visualizzazione di una linea verso il basso), visualizzando una linea vuota all'estremo inferiore dello schermo; premendo **Ctrl+Iniz**, il cursore ritorna alla prima linea del programma. QuickBASIC permette linee contenenti sino a 255 caratteri e, se un programma contiene linee che oltrepassano tale limite sarà possibile far scorrere la finestra sulla destra premendo il tasto **Freccia Destra**. Per riportare la finestra alla sua posizione originale, si prema il tasto **Freccia Sinistra**.

2.5.2 Memorizzazione di un programma GW-BASIC

Si inserisca il disco dati di QuickBASIC contenente il Programma 2.1 nell'unità B in modo da poter memorizzarci il Programma 1.3: si selezioni il comando Salva con nome del menu

```
File  Modifica  Visualizza  Ricerca  Esegui  Debug  Opzioni                Guida
                              UQB0103.BAS
1 REM ** VALORE DEI TITOLI **
2 ' &TTL, Capitolo 1
3 ' GW-BASIC  File: UQB0103.BAS
9 '
100 REM ** IMMISSIONE DATI E CALCOLO DEL VALORE **
110 CLS : WORTH# = 0: SHARES = .0001
120 ST$ = "$$###,###,###.##"
130 WHILE SHARES <> 0
140   INPUT "NUMERO DI AZIONI  "; SHARES
150   INPUT "PREZZO PER AZIONE "; PPS
160   WORTH# = WORTH# + SHARES * PPS
170   PRINT
180 WEND
199 '
200 REM * ARROTONDA E STAMPA IL VALORE **
210 WORTH# = INT(WORTH# * 100 + .5) / 100
220 PRINT "VALORE TOTALE DEI TITOLI = "; : PRINT USING ST$; WORTH#
230 END
                              Immediata
<MAIUSC+F1=Guida> <F6=Fin.> <F2=SUB> <F5=Esegui> <F8=Passo>      M  00001:001
```

Figura 2.13 Ambiente schermo con UQB0103.BAS caricato

File; poiché il programma, quando è stato caricato, aveva già un nome, questo compare nella casella "Nome del file:" con acclusa la designazione dell'unità ad esso relativa. Si prema quindi semplicemente il tasto **Invio**. Dopo aver memorizzato il programma si acceda al menu File, si evidenzi il comando Apri programma, si prema **Invio** e, quando compare il riquadro di comunicazione, si digiti nella casella "Nome del file:":

```
B:*.BAS
```

L'asterisco (carattere jolly) chiede al calcolatore di elencare tutti i file con estensione .BAS contenuti nel dischetto nell'unità B; premendo **Invio**, vengono elencati i seguenti file:

```
UQB0103.BAS
UQB0201.BAS
```

Si prema il tasto **Esc** per eliminare il riquadro di comunicazione e riportarsi alla finestra divisualizzazione: si è ora pronti ad eseguire il programma.

2.5.3 Esecuzione di un programma GW-BASIC

Si prema la combinazione di tasti **Alt+E** per accedere al comando Avvia del menu Esegui ed **Invio** per eseguire il programma. Avendo già eseguito il Programma 1.3 da GW-BASIC

e poiché il programma non contiene comandi proibiti, ci si potrà attendere che funzioni con successo in QuickBASIC.

Il programma funzionerà come si è già visto nel Capitolo 1: si inserisca, come si era fatto precedentemente, il numero di azioni ed il prezzo di ogni azione per ogni titolo e, quando si digiterà **0** come numero di azioni ed un qualunque numero come prezzo, verrà visualizzato il valore del portafoglio titoli.

2.6 Altri Menu

Si è visto come si presentano i menu File ed Esegui ed il loro funzionamento, mentre viene ora presentata una breve descrizione delle informazioni che appaiono sulla barra di riferimento quando si selezionano gli altri menu del sistema semplificato.

Questi menu verranno trattati in maggior dettaglio quando saranno utilizzati nel seguito del libro.

2.6.1 Menu Modifica

Il menu Modifica contiene tre comandi: Taglia, Copia ed Incolla, utilizzati per modificare i programmi.

Taglia	Cancella il testo selezionato copiandolo nel buffer.
Copia	Copia il testo selezionato nel buffer.
Incolla	Inserisce il contenuto del buffer alla posizione corrente.

2.6.2 Menu Visualizza

Il menu Visualizza viene utilizzato per visualizzare le parti del programma nella finestra di visualizzazione o per visualizzare lo schermo di uscita.

SUBroutine	Visualizza SUBroutine, FUNCTION, modulo, file da includere o documento.
Schermo output	Visualizza lo schermo dell'output.
Righe incluse	Visualizza il file da includere senza permetterne la modifica.

2.6.3 Menu Ricerca

Il menu Ricerca viene utilizzato per la ricerca o la sostituzione di un testo specificato.

Trova	Trova il testo specificato.
Sostituisci	Trova e cambia il testo specificato.

2.6.4 Menu Debug

Il menu Debug viene utilizzato per controllare l'esecuzione di un programma con strumenti sofisticati per la correzione (Debugging) che permettono di controllare espressioni o variabili particolari o di interrompere il programma in punti specificati.

Osserva	Aggiunge l'espressione specificata alla finestra di osservazione.
Osservazione immediata	Visualizza il valore di una variabile od espressione.
Elimina dall'osservazione	Elimina la voce selezionata dalla finestra di osservazione.
Punto di interruzione	Attiva o disattiva un punto di interruzione sul cursore.
Elimina ogni punto di osservazione	Elimina tutti i punti di osservazione.

2.6.5 Menu Opzioni

Il menu Opzioni permette di selezionare proprietà specificate dell'ambiente QuickBASIC.

Display	Cambia i parametri per la visualizzazione.
Percorsi di ricerca	Imposta i percorsi di ricerca predefiniti.
Menu completi	Alterna tra le opzioni Menu brevi e menu completi.

2.6.6 Menu Guida

Il menu Guida viene utilizzato per ricevere assistenza con informazioni specifiche.

Indice	Visualizza l'indice della Guida.
Panoramica	Visualizza il sommario della Guida.
Argomento:	Visualizza i dati sulla parola chiave di BASIC su cui si trova il cursore.
Uso della Guida	Visualizza informazioni sull'uso della Guida.

2.7 Sommario

In questo capitolo sono state fatte le copie di sicurezza dei dischetti originali di QuickBASIC, è stato formattato un dischetto vuoto come disco di sistema ed è stato copiato su questo dischetto il dischetto dei programmi. Il risultato è stato un dischetto di lavoro QB che si è utilizzato per scrivere, memorizzare ed eseguire un programma QuickBASIC nuovo, LANCIO MONETE, per caricare ed eseguire un programma QuickBASIC precedentemente memorizzato e per caricare, memorizzare ed eseguire un programma GW-BASIC, VALORE DEI TITOLI, che era stato memorizzato in formato ASCII.

Si è appreso come utilizzare i menu File ed Esegui per creare un nuovo programma, per richiamarne uno precedentemente memorizzato ed inoltre ad eseguire entrambe i tipi di programma; si è poi appreso come rispondere ai riquadri di comunicazione che compaiono quando il comando di un menu richiede maggiori informazioni. Sono stati elencati i comandi dei menu Modifica, Visualizza, Ricerca, Debug, Opzioni e Guida con una breve descrizione.

Capitolo 3

Realizzazione di strumenti di lavoro

3.1 Creazione di un insieme di strumenti
3.2 Variabili e tipi di dati
3.3 Linee di programma
3.4 Funzioni predefinite
3.5 Funzioni definite dall'utente
3.6 Dischetto strumenti
3.7 Definizione di funzioni
3.8 Funzioni su più linee
3.9 Strutture di controllo
3.10 Sommario

Ogni linguaggio possiede il proprio insieme di regole che devono essere rispettate per poter effettivamente comunicare: in questo capitolo verranno trattate alcune delle più importanti da seguire quando si utilizza QuickBASIC; come le stringhe e le variabili lunghe, i blocchi di istruzioni, i formati strutturati e le SUBroutine e FUNCTION.

Questo capitolo ha inizio con la realizzazione di strumenti di programmazione che potranno essere utilizzati nei futuri programmi; memorizzando uno strumento su disco come file separato è possibile inglobarlo, quando necessario, nei propri programmi. Gran parte del capitolo è dedicata a funzioni su una sola linea, oppure su più linee: variabili locali e globali ed operatori logici (Booleane).

Vengono introdotte, descritte ed utilizzate tre strutture di controllo di QuickBASIC: i blocchi IF...THEN...ELSE, SELECT...CASE e DO...LOOP. Il blocco con la struttura IF..THEN...ELSE estende il comando su singola linea di GW-BASIC IF...THEN...ELSE in un blocco su più linee, rendendo la struttura più leggibile e consentendo maggiore versatilità; la struttura SELECT...CASE permette scelte multiple a seconda del valore di un'espressione, numerica o stringa; la struttura DO...LOOP permette di controllare indif-

ferentemente al principio od al termine di un ciclo se una condizione è vera o falsa ed il ciclo verrà eseguito mentre (WHILE) la condizione è soddisfatta o sino a che (UNTIL) la condizione non sia soddisfatta. Questa struttura permette l'eliminazione delle istruzioni GOTO che spesso creano confusione.

3.1 Creazione di un insieme di strumenti

Nell'apprendere l'uso di QuickBASIC si dovrebbero raccogliere strumenti riutilizzabili in futuro, come le funzioni su più linee e su singola linea introdotte in questo capitolo per creare un ambiente di lavoro, risparmiando così del tempo e scrivendo programmi in modo più efficiente. Gli strumenti di programmazione potrebbero essere delle definizioni di funzioni, memorizzate come file separati su un solo dischetto che potranno facilmente essere aggiunte a qualsiasi programma con il comando Merge del menu File (utilizzando il sistema di menu completo). Quando è necessaria una funzione in un programma si operi come segue: si inserisca nell'unità B il dischetto contenente le definizioni di funzione, si sposti il cursore nel punto in cui si desidera inserire la definizione, si selezioni il comando Merge nel menu File, quando compare la finestra di dialogo si digiti il nome del file della funzione nella casella Nome del file e si prema **Invio**; in pochi attimi la definizione verrà aggiunta al programma. Questa procedura verrà descritta in dettaglio nel prossimo capitolo.

Si dovrebbero ponderare con particolare attenzione i nomi da dare ai file contenenti le definizioni di funzione: essi dovrebbero infatti descrivere ciò che la funzione compie, pur sottostando alla restrizione del DOS che impone un nome di non più di otto caratteri ed un'estensione di non più di tre. Se il file desiderato porta il medesimo nome della funzione sarà più semplice da individuare; si tratterà ad esempio in seguito una funzione su singola linea che arrotonderà una quantità di denaro al più vicino centesimo, chiamata FNroundcnt: il nome appropriato per il file sarebbe quindi ROUNDCNT, poiché il nome descrive la funzione (ROUND, arrotonda, al più vicino CNT) divenendo semplice localizzarla in un elenco di file.

3.2 Variabili e tipi di dati

In QuickBASIC esistono due tipi di variabili: stringa e numeriche, ciascuna delle quali deve corrispondere sempre all'espressione che le è assegnata: tipi stringa con stringhe e tipi numerici con numeri.

- citta$ = "MILANO" (sia citta$ che "MILANO" sono stringhe)
- cap = 20100 (sia cap che 20100 sono numerici)

I nomi di variabile in QuickBASIC possono contenere sino a 40 caratteri: sono permessi lettere, numeri, punti decimali e caratteri dichiaratori di tipo ($ % & ! #) con il vincolo che il primo carattere sia una lettera; una variabile non può essere una parola riservata (parola chiave), mentre queste possono essere incluse in una variabile. Ad esempio, FOR, che è una parola chiave, è inclusa in Forza, che è una variabile e non una parola riservata; è anche accettabile il nome di variabile FOR$, poiché il carattere di dichiarazione ($) rende FOR incluso in un nome completo, anche se sarebbe tuttavia da evitare.

Le lettere in un nome di variabile possono essere maiuscole o minuscole: CAP, Cap e cap sono tutte la stessa variabile; se si prova ad utilizzare la variabile CAP in una parte iniziale del programma ed in seguito Cap, QuickBASIC modificherà tutte le precedenti variabili nell'ultima forma utilizzata: in questo caso tutte le variabili CAP digitate precedentemente verranno trasformate in Cap.

Questa caratteristica è molto utile quando si decide di modificare la forma di una variabile (maiuscole o minuscole) dopo che un programma (o parte di un programma) sia già stato scritto; sarà infatti sufficiente effettuare la modifica in un solo punto e QuickBASIC effettuerà la modifica in tutti gli altri.

Da questo momento in avanti le variabili compariranno in questo libro in una delle seguenti forme:

- tutto minuscolo: cap citta$ anno% soldi#;
- prima lettera maiuscola: Agente007$ Cap;
- lettere selezionate maiuscole: CodiceCap$ GrandeNumero#.

Per agevolare la distinzione tra variabili e parole chiave, che compaiono in sole maiuscole, le variabili non compariranno in sole lettere maiuscole.

Vengono utilizzati cinque caratteri speciali per specificare le variabili come stringa ($), intero corto (%), intero lungo (&), singola precisione (!), doppia precisione (#).

**stato$** Il simbolo di dollaro specifica che la variabile stato$ rappresenta una stringa di caratteri: possono essere utilizzate stringhe sino a 32767 caratteri.

calorie% Il simbolo di percentuale specifica che la variabile calorie% rappresenta un intero: gli interi sono valori numerici memorizzati come interi a 16-bit nell'intervallo da -32768 a 32767.

centesimi& La e commerciale specifica che la variabile centesimi& rappresenta un intero lungo, memorizzato come intero a 32-bit nell'intervallo da -2147483648 a 2147483647.

PrezzoMedio! Il punto esclamativo specifica che la variabile PrezzoMedio! rappresenta un numero in virgola mobile (floating-point) a singola precisione, che è cioè preciso sino circa alla settima cifra decimale ed il suo campo è da -3.4E+38 a -1.4E-45 per i numeri negativi, zero e per i numeri positivi da 1.4E-45 a 3.4E+38.

Moltisoldi# Il simbolo di numero specifica che la variabile Moltisoldi# rappresenta un numero floating-point in doppia precisione, cioè preciso circa sino alla quindicesima cifra decimale ed il cui campo va da -1.8D+308 a -4.9D-324 per i numeri negativi, zero e per i numeri positivi da 4.9D-324 a 1.8D+308. Nei numeri in doppia precisione viene utilizzata per designare la potenza la lettera D anziché la E.

Il tipo di default per una variabile, quando non viene specificato $, %, &, ! o #, è il floating-point in doppia precisione, sempre che non si definisca il tipo di default con l'istruzione DEFtipo (DEFINT, DEFSNG, DEFDBL, DEFLNG o DEFSTR). È anche possibile dichiarare un tipo di variabile in una proposizione AS in uno qualunque dei seguenti comandi: DIM, COMMON, REDIM, SHARED o STATIC. Le specifiche dei tipi di dati sopra elencate possono essere utilizzate per risparmiare memoria; vi sono inoltre differenze nelle velocità di esecuzione tra i vari tipi di dati, immagazzinati nel seguente modo:

Intero corto (%)	2 byte
Intero lungo (&)	4 byte
Singola precisione (!)	4 byte
Doppia precisione (#)	8 byte
Stringhe ($)	6 byte per stringa più un byte per ogni carattere

Utilizzare quando possibile interi corti rende ovviamente possibile risparmiare grandi quantità di memoria. Vi sono due utili operatori associati agli interi: la barra inversa (\) e l'operatore MOD. La barra inversa opera le divisioni tra interi, arrotondando dividendo e divisore all'intero più vicino e, ad operazione compiuta, troncando (ovvero eliminando la parte decimale) il risultato e generando quindi un quoziente intero. Ad esempio:

20\3=6

20.49\3.1=6 (come 20\3=6)

20.51\3.1=7 (come 21\3=7)

19.49\3.6=4 (come 19\4=4)

19.51\3.6=5 (come 20\4=5)

L'operatore MOD calcola il resto di una divisione intera, ad esempio:

20 MOD 3 = 2

20 MOD 4 = 0

20 MOD 5 = 0

1987 MOD 4 = 3

Questo operazione è anche chiamata modulo e l'operatore MOD è chiamato operatore modulo.

3.3 Linee di programma

Una linea di programma può contenere un massimo di 255 caratteri, spazi inclusi, e viene conclusa premendo il tasto **Invio**; se si digita una linea contenente più di 78 caratteri la finestra di visualizzazione scorre a destra dopo che si sono digitati i primi 78 caratteri (si pensi anche alla linea che si sposta a sinistra) e, quando si preme **Invio** al termine della linea, torna alla propria posizione originale. Nella finestra di visualizzazione è possibile vedere solamente 78 caratteri contemporaneamente: se si desidera vedere oltre il margine destro della finestra, si tenga premuto il tasto **Freccia Destra** per spostare la finestra a destra ed il tasto **Freccia Sinistra** per riportarsi all'inizio della linea.

Poiché è scomodo leggere un programma sullo schermo se contiene linee più lunghe della finestra e poiché inoltre, stampando il programma con una stampante ad 80 colonne, le linee più lunghe di 80 caratteri verranno spezzate in due o più linee, generalmente in punti scomodi e producendo un listato di difficile lettura che non assomiglia esattamente al programma visualizzato sullo schermo, per ottenere un buon stile di programmazione è fortemente raccomandato l'uso di linee non più lunghe di 80 caratteri. D'ora innanzi, i programmi presenti in questo libro seguiranno questa raccomandazione.

Come si è visto, non è necessaria la numerazione delle linee, ma, se in una particolare occasione una linea deve essere identificata, è possibile utilizzare sia un numero di linea, sia una etichetta alfanumerica. I numeri di linea sono identificatori come quelli utilizzati in GW-BASIC, con un campo compreso tra 0 e 65529, ma, poiché in QuickBASIC non sono necessari, in questo libro vengono utilizzati solo quando si tratta di programmi GW-BASIC o quando questi vengono confrontati con quelli in QuickBASIC.

Un'etichetta alfanumerica può essere costituita da una combinazione pari ad una quantità di lettere o cifre compresa tra 1 e 40, terminante con un carattere due punti; non sono permesse come etichette parole chiave di QuickBASIC, eccezione fatta nel caso in cui queste siano incluse nell'etichetta; sono accettate indifferentemente lettere maiuscole o minuscole: INIZIAQUI:, iniziaqui:, IniziaQui ed Iniziaqui sono la medesima etichetta.

Le istruzioni GOTO, che richiedono l'uso di numeri di linea o di etichette di linea, saranno evitate in questo libro ovunque sia possibile, ma in ogni caso vengono inclusi alcuni esempi sull'uso di etichette di linea con queste istruzioni. In un'istruzione che rimanda ad una etichetta di linea, viene omesso il carattere due punti al termine dell'etichetta, che va tuttavia incluso al termine della linea a cui si riferisce, come qui illustrato:

```
GOTO inizio
.
.
inizio:
FOR numero = 1 TO 25
  car$ = MID$(stringa$, k, 1)
  IF car$ <> " " THEN nospazi$ = nospazi$ + car$
NEXT numero
```

Quando si utilizzano etichette di linea, un'istruzione IF...THEN ha la seguente forma:

```
IF saldo > 0 THEN GOTO VaBene
.
.
VaBene:
```

3.4 Funzioni predefinite

QuickBASIC ha un ricco repertorio di funzioni predefinite; una funzione opera su uno o più argomenti e produce o restituisce un valore: gli argomenti possono essere numerici o stringhe ed una funzione può essere una funzione numerica od una funzione stringa; una funzione numerica restituisce un valore numerico, mentre una funzione stringa restituisce un valore stringa. Seguono alcuni esempi dei due tipi di funzione.

ASC INSTR INT LEN RND VAL	Funzione numerica
CHR$ MID$ SPACE$ STR$ STRING$	Funzione stringa

Il nome di una funzione stringa termina sempre con il simbolo di dollaro. Sono possibili molte combinazioni di funzioni ed argomenti.

INT(100 * soldi + .5)	Funzione numerica, argomento numerico.
LEN(parola$)	Funzione numerica, argomento stringa.
STR$(codicecap)	Funzione stringa, argomento numerico.
UCASE$(STATO$)	Funzione stringa, argomento stringa.
INSTR(inizio, stringa$, sottostringa$)	Funzione numerica, argomento misto.
MID$(parola$, posizione, 1)	Funzione stringa, argomento misto.

QuickBASIC mette a disposizione funzioni inesistenti in Applesoft, BASIC e GW-BASIC, incluse le utili LCASE$ e UCASE$. LCASE$(stringa$) è una funzione stringa di un argomento stringa che restituisce il valore dell'argomento (stringa$) con tutte le lettere maiuscole trasformate in minuscole; i caratteri che non sono lettere restano immutati.

Anche UCASE$(stringa$) è una funzione stringa di un argomento stringa, ma restituisce il valore dell'argomento (stringa$) con tutte le lettere minuscole trasformate in maiuscole ed i caratteri non alfabetici immutati.

Se il valore originale di stringa$ fosse LOMBARDIA, il valore reso da LCASE$(stringa$) sarebbe lombardia.

Verranno in seguito introdotte altre funzioni di QuickBASIC non disponibili in GW-BASIC.

3.5 Funzioni definite dall'utente

È possibile utilizzare l'istruzione DEF FN per definire una propria funzione ed assegnarle un nome: QuickBASIC permette di definire funzioni su singola linea esattamente come in GW- BASIC od Applesoft, ma consente anche la definizione di funzioni su più linee.

Si designa e definisce una funzione su singola linea ponendo il nome della stessa immediatamente dopo DEF FN, come in DEF FNroundcnt#; se il nome è una variabile numerica, la funzione è una funzione numerica, mentre, se il nome è una variabile stringa, la funzione è stringa: DEF FNlanci$ ne è un esempio. Può seguire un elenco di nomi di variabili separate da virgole (detta elenco di parametri): queste variabili sono variabili fittizie che non hanno alcun significato al di fuori della definizione di funzione ed è dunque possibile utilizzare nomi di variabili già utilizzate nel programma. La definizione di funzione è separata dal suo nome con un segno di uguale ed è l'espressione che calcola il valore, stringa o numerico, assegnato alla funzione, che deve essere dello stesso tipo (stringa o numerico) della variabile che dà il nome alla funzione. Ad esempio:

```
DEF FNroundcnt# (m#) = SGN(m#) * INT(100 * abs(m#) + .5) / 100
DEF FNlanci$ = MID$("TESTACROCE",5 * INT(2 * RND) + 1, 4)
```

Una funzione DEF FN deve essere definita prima di essere chiamata o utilizzata, si dovrebbe dunque porre le definizioni nella parte iniziale del programma in cui vengono utilizzate.

```
REM ** DOLLARI E CENT **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0301.BAS

REM ** Definisce la funzione FNroundcnt# **
' Funzione in doppia precisione per arrotondare
' un numero in doppia precisione a due decimali
DEF FNroundcnt# (m#) = SGN(m#) * INT(100 * ABS(m#) + .5) / 100

REM ** PROGRAMMA PRINCIPALE PER PROVARE LA FUNZIONE **
' Pulisce lo schermo poi chiama la funzione
' Premere il tasto Esc per terminare
CLS
DO
   INPUT "Numero, prego"; dato#
   PRINT "Il valore arrotondato e'"; FNroundcnt#(dato#)
   PRINT
   PRINT "Premere Esc per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END
```

Programma 3.1 DOLLARI E CENT

Il Programma 3.1, DOLLARI E CENT, illustra la definizione di una funzione numerica di una variabile numerica per arrotondare un numero al centesimo più vicino; la funzione, FNroundcnt#, è definita come funzione in doppia precisione in modo da poter essere calcolata con 15 cifre decimali, mentre la seguente versione in singola precisione è limitata a 7 cifre decimali:

```
DEF FNroundcnt(m) = SGN(m) * INT(100 * ABS(m) + .5) / 100
```

Entrambe queste funzioni sono utili per arrotondare i numeri nei propri programmi: si utilizzi l'istruzione PRINT USING per dare un formato ai numeri quando vengono stampati.

Poiché i numeri vengono internamente immagazzinati in formato binario, la rappresentazione binaria, per i non interi, potrebbe non essere esattamente coincidente con quella decimale; quindi occasionalmente una funzione semplice come FNroundcnt# potrebbe non arrotondare correttamente un numero il cui valore decimale abbia solamente tre cifre decimali e la terza sia 5.

Sarà possibile osservare che in molte definizioni di funzioni su singola linea verranno utilizzati nomi di variabili brevi; essendo infatti le definizioni di funzioni su singola linea sovente molto lunghe, in queste sono spesso utilizzati nomi di variabili brevi in modo da non oltrepassare il termine della linea.

Il programma DOLLARI E CENT illustra lo stile di programmazione che verrà utilizzato in tutto il libro: ogni blocco ha inizio con un'istruzione REM ed altre istruzioni esplicative sono precedute da un'apostrofo ('), la forma abbreviata di REMARK; questo stile rende più agevole l'individuazione e la lettura delle parti funzionali separate del programma.

La Figura 3.1 illustra una esecuzione di prova del programma: si osservi che i numeri sono correttamente arrotondati a due cifre decimali; dopo la stampa del numero arrotondato vi è una opzione EXIT DO: si prema il tasto **Esc** per abbandonare il ciclo DO...LOOP e terminare il programma e si prema quindi un qualsiasi altro tasto per arrotondare un nuovo numero.

```
Numero, prego? 123.456
Il valore arrotondato e' 123.46

Premere Esc per finire od un altro tasto per continuare
Numero, prego? 123.455
Il valore arrotondato e' 123.45

Premere Esc per finire od un altro tasto per continuare
Numero, prego? -123.456
Il valore arrotondato e' -123.46

Premere Esc per finire od un altro tasto per continuare
```

```
Numero, prego? 123456789.946
Il valore arrotondato e' 123456789.95

Premere Esc per finire od un altro tasto per continuare
```

Figura 3.1 Output del Programma 3.1

3.6 Dischetto strumenti

È possibile incominciare a creare il proprio dischetto di strumenti di programmazione memorizzando in un file la funzione FNroundcnt, senza salvare l'intero programma DOLLARI E CENT: si cancelli tutto nel programma eccetto la definizione di funzione e le note descrittive:

```
REM ** Definisce la funzione FNroundcnt# **
' Funzione in doppia precisione per arrotondare
' un numero in doppia precisione a due decimali
DEF FNroundcnt# (m#) = SGN(m#) * INT(100 * ABS(m#) + .5) / 100
```

Per memorizzare la funzione si acceda al comando Salva con nome del menu File e si digiti **B:ROUNDCNT** come nome del file nella casella nome del file e, poiché in seguito si incorporerà la definizione di funzione in un programma principale, la si memorizzi in formato testo.

Si osservi il riquadro di comunicazione Formato, se è evidenziato "QuickBASIC-Carica e salva velocemente", si prema il tasto **Tab** per spostarsi nella casella Formato, si prema quindi il tasto **Freccia Giù** per evidenziare "Testo-Leggibile ad altri programmi" e, se non è evidenziata la casella OK, si prema il tasto **Tab** per evidenziarla, come illustrato nella Figura 3.2; si prema infine il tasto **Invio**.

Volendo verificare se la definizione di funzione sia stata memorizzata, si acceda al riquadro di comunicazione Apri programma del menu File e si esamini la directory del dischetto nell'unità B digitando **B:*.*** come nome di file.

```
Nome del file: B:*.*
```

Si prema quindi **Invio** per vedere i nomi di tutti i file sul dischetto nell'unità B: poiché ROUNDCNT è il primo file sul dischetto di strumenti, questo dovrebbe essere l'unico nome di file elencato.

Nel Capitolo 4, si apprenderà come includere in altri programmi una definizione di funzione memorizzata su disco.

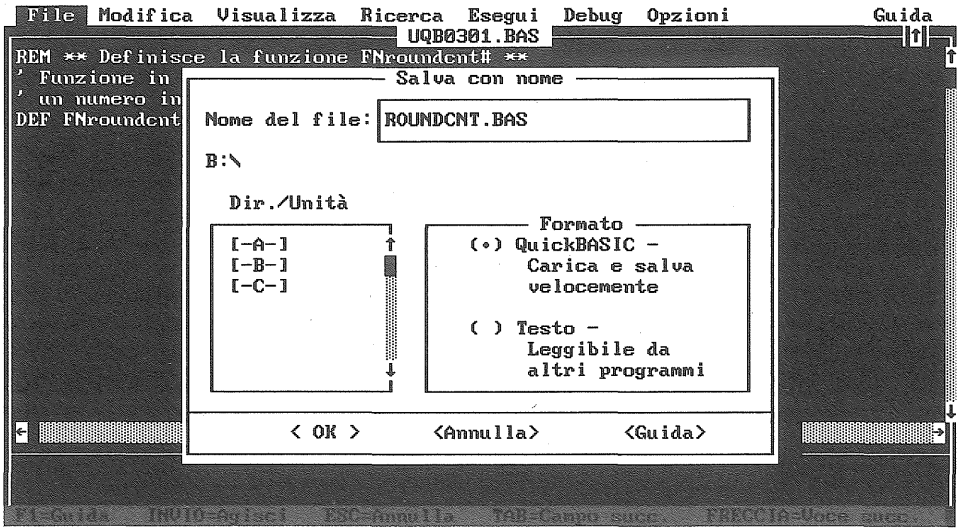


Figura 3.2 Riquadro di comunicazione di Salva con nome con Testo selezionato

3.7 Definizione di funzioni

Una funzione numerica può essere un intero corto, intero lungo, singola precisione, doppia precisione a seconda della variabile utilizzata per designarla. Il seguente elenco illustra i quattro metodi per definire un funzione che restituisce un numero compreso tra 1 ed n, dove n è un numero positivo.

```
DEF FNintcas% (n%) = INT(n% * RND) + 1  'Funzione intero corto
DEF FNintcas& (n&) = INT(n& * RND) + 1  'Funzione intero lungo
DEF FNintcas (n) = INT(n * RND) + 1  'Funzione in singola precisione
DEF FNintcas# (n#) = INT(n# * RND) + 1  'Funzione in doppia precisione
```

Si selezioni la DEF FN con l'identificatore adeguato alle proprie necessità: si ricordi che il campo di FNintcas%(n%) è da 1 a 32767.

3.7.1 Funzioni Massimo e Minimo

Una funzione numerica può essere definita in modo tale da restituire il massimo od il minimo di due numeri:

```
DEF FNmin (a,b) = a * ABS(a <= b) + b * ABS(a > b)
```

Come funziona? Si supponga che il valore di a sia 4 e quello di b 7, in questo caso il valore di a<=b è 1 (vero) ed il valore di a>b è 0 (falso), quindi il valore dell'intera espressione è a.

```
a * ABS(a <= b) + b * ABS(a > b) = a
```

Il valore di FNmin è:

```
a * 1 + b * 0
```

Si supponga ora che a valga 6 e b 3, in questo caso:

```
a * ABS(a <= b) + b * ABS(a > b) = b
```

Il valore di FNmin è:

```
a * 0 + b * 1 = b
```

Se a e b hanno uguale valore viene restituito il valore di a.

Una funzione simile, FNmax, restituisce il massimo di due numeri:

```
DEF FNmax(a,b) = a * ABS(a >= b) + b * ABS(a < b)
```

Il Programma 3.2, MINIMO E MASSIMO, illustra entrambe le funzioni: se si pensa che queste possano essere utili, le si memorizzino entrambe in file distinti sul dischetto di strumenti; si potrebbe anche scrivere e memorizzare di entrambe le funzioni le versioni intero corto, intero lungo ed in doppia precisione.

Si scelgano nomi e nomi di file che rendano facile l'individuazione della funzione desiderata, ad esempio:

- FNMININT per la funzione intero corto FNmin%
- FNMINLNG per la funzione intero lungo FNmin&
- FNMINSGL per la funzione in singola precisione FNmin
- FNMINDBL per la funzione FNmin# in doppia precisione

```
REM ** MINIMO E MASSIMO **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0302.BAS

REM ** Definizioni delle funzioni **
' FNmin e' una funzione su singola linea che restituisce
' il minimo di due numeri in singola precisione
DEF FNmin (a, b) = a * ABS(a <= b) + b * ABS(a > b)

' FNmax e' una funzione su singola linea che restituisce
' il massimo di due numeri in singola precisione
DEF FNmax (a, b) = a * ABS(a >= b) + b * ABS(a < b)

REM ** Programma principale per provare le funzioni **
CLS
DO
```

```
    INPUT "Primo numero"; primo
    INPUT "Secondo numero"; secondo
    PRINT "Il minimo e'"; FNmin(primo, secondo)
    PRINT "Il massimo e'"; FNmax(primo, secondo)
    PRINT
    PRINT "Premere Esc per finire od un altro tasto per continuare"
    ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END
```

Programma 3.2 MINIMO E MASSIMO

Si dovrebbe anche stampare la directory del dischetto degli strumenti ed inserirlo nella custodia insieme al dischetto ogniqualvolta si aggiungano nuovi file.

3.7.2 Definizione di funzioni senza parametri

È possibile definire funzioni senza specificare un elenco di parametri, come negli esempi che seguono.

- La funzione FNlancia3D6% simula il lancio di tre dadi a sei facce (3D6) e restituisce un intero compreso fra 3 e 18.

  ```
  DEF FNlancia3D6% = INT(6 * RND) + INT(6 * RND) + INT(6 * RND) + 3
  ```

- La funzione FNlanci$ simula il lancio di una moneta e restituisce il valore stringa TESTA o CROCE, scelto a caso.

  ```
  DEF fnlanci$ = MID$("TESTACROCE", 5 * INT(2 * RND) + 1, 5)
  ```

- La funzione FNvocalecas$ restituisce una vocale a caso.

  ```
  DEF FNvocalecas$ = MID$("aeiou"), INT(5 * RND) + 1, 1)
  ```

3.7.3 Utilizzo di definizioni di funzione

Seguono due programmi che illustrano l'uso di definizioni di funzione: il primo programma determina se un anno è bisestile o meno, il secondo genera parole di cinque lettere a caso.

```
REM ** ANNO BISESTILE **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0303.BAS

REM ** Definisce le Variabili, poi le Funzioni **
DEFINT A-Z
```

```
' FNbis% ™ una funzione su singola linea che restituisce
' -1 se l'argomento e' un anno bisestile
'  0 se l'argomento e' un anno comune
DEF FNbis% (a) = (a MOD 4 = 0 AND a MOD 100 <> 0) OR a MOD 400 = 0

REM ** Programma principale **
CLS
DO
   INPUT "Anno"; year
   IF FNbin%(anno) THEN PRINT "Anno bisestile" ELSE PRINT "Anno normale"
   PRINT
   PRINT "Premere Esc per finire od un altro tasto per continuare"
LOOP
END
```

Programma 3.3 ANNO BISESTILE

Un anno, secondo il calendario Gregoriano introdotto nel 1582, può essere normale (365 giorni) o bisestile: un anno è bisestile quando divisibile per 4, eccettuati i centennali. I centennali sono gli anni multipli di 100 (1800, 1900, 2000 e così via) e sono bisestili solamente se multipli di 400. L'anno 2000 è bisestile, mentre il 1900 non lo è. Un anno è dunque bisestile se è divisibile per quattro e non è un centennale, oppure se è un centennale ed è divisibile per 400; questa definizione viene utilizzata nella funzione FNbis%, illustrata nel Programma 3.3, ANNO BISESTILE.

Nella Figura 3.3 è illustrata una esecuzione di prova del programma, il quale sembra funzionare correttamente; è tuttavia sempre consigliabile una prova più approfondita.

```
Anno? 1502
Anno normale

Premere Esc per finire od un altro tasto per continuare
Anno? 1988
Anno bisestile

Premere Esc per finire od un altro tasto per continuare
Anno? 1900
Anno normale

Premere Esc per finire od un altro tasto per continuare
Anno? 1990
Anno normale
```

```
Premere Esc per finire od un altro tasto per continuare
Anno? 2000
Anno bisestile

Premere Esc per finire od un altro tasto per continuare
```

Figura 3.3 Output del Programma 3.3

La funzione FNbis% restituisce il valore -1 se l'argomento è un anno bisestile od altrimenti il valore 0, per cui è essenzialmente una funzione logica o Booleana che restituisce i valori vero (-1) o falso (0). La Tabella 3.1 illustra come il valore di FNbis% sia determinato per alcuni valori dell'argomento; le lettere A, B e C sono utilizzate nel modo seguente:

A: anno MOD 4 = 0

B: anno MOD 100 <> 0

C: anno MOD 400 = 0

Tabella 3.1 Calcolo del valore di FNbis%

Anno	A	B	A and B	C	(A and B) or C	FNbis%
1900	-1	0	0	0	0	0
1987	0	-1	0	0	0	0
1988	-1	-1	-1	0	-1	-1
2000	-1	0	0	-1	-1	-1

I calcolatori possono inventare parole: il Programma 3.4, INVENTAPAROLE NUMERO 1, genera combinazioni casuali di cinque lettere nella forma di consonante, vocale, consonante, vocale, consonante. Gran parte del lavoro è eseguito dalla funzione stringa FNcaschr$, che sceglie a caso un carattere da una stringa di consonanti o vocali.

```
REM ** INVENTAPAROLE NUMERO 1 **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0304.BAS

REM ** Definizione di funzione **
' FNcaschr$ e' una funzione su singola linea che restituisce
' un carattere casuale da una stringa
DEF FNcaschr$ (stringa$) = MID$(stringa$, INT(LEN(stringa$) * RND) +1, 1)
REM ** Programma principale **
' Definisce le variabili
CLS : RANDOMIZE TIMER: DEFINT A-Z
consonanti$ = "BCDFGHJKLMNPQRSTVWXYZ"
```

```
vocali$ = "AEIOUY"
parola$ = ""
' Forma e stampa parole
FOR k = 1 TO 80
   parola$ = FNcaschr$(consonanti$)
   parola$ = parola$ + FNcaschr$(vocali$)
   parola$ = parola$ + FNcaschr$(consonanti$)
   parola$ = parola$ + FNcaschr$(vocali$)
   parola$ = parola$ + FNcaschr$(consonanti$)
   PRINT parola$,
NEXT k
END
```

Programma 3.4 INVENTAPAROLE NUMERO 1

Nella Figura 3.4 è illustrata una esecuzione di prova: ogni parola ha la forma c,v,c,v,c, dove c è una consonante e v una vocale; la lettera y può comparire in qualunque posizione poiché è contenuta in entrambe le stringhe (consonanti$ e vocali$). La prossima volta che si desidera inventare il nome di una società, di un prodotto o di un personaggio di un romanzo che si sta scrivendo, si provi ad utilizzare FNcaschr$. È facile scrivere un programma che generi parole casuali con la desiderata struttura consonanti-vocali modificando semplicemente l'istruzione di assegnamento utilizzata per determinare il valore di parola$.

```
XEBIW    XUPUY    YAPIM    MOCIG    VUHIW
KYFIJ    TYMID    ZEXOM    MAZIY    RIRAB
WAWYB    FEKAG    PODAV    XUMER    JYBUM
REQOV    VEBYL    CEFUJ    ROYER    WALIY
QOYYT    VIGOH    JIGOP    WUMAC    CEQIH
BITEG    HEWYU    SUQUD    MUHAL    PECOV
HIKOM    TUVUZ    XOTYH    WYTAM    HADUW
HEPYT    GUPUY    MIDIS    YYPYS    QERYL
PAWAZ    POKUK    RUZOM    KOJEM    KYMAD
WABYK    LEBAM    BEGYT    LOJAF    POWAX
MUXYI    QYZIH    QETIH    FEVIW    JYDUZ
SAREP    GUSUJ    SYDAB    LARIP    SYXEY
MUPET    WUJIC    GYTUF    MYSUB    ZOLIJ
XEGAZ    JYBOM    DAXEW    PATOS    KIMOX

Premere un tasto per continuare
```

Figura 3.4 Output del Programma 3.4

In conclusione: la definizione di funzione su singola linea DEF FN esegue l'operazione alla destra dell'uguale, è limitata ad una singola linea di codice ed i nomi di variabile che compaiono nella espressione sono locali all'espressione stessa e sono utilizzati per definire la funzione senza avere ripercussioni sulle variabili del programma con il medesimo nome (sono variabili fittizie). Se un nome di variabile compare in un elenco di parametri, il parametro viene fornito quando si richiama la funzione dal programma, mentre, se una variabile non compare nell'elenco di parametri, viene trattata come una variabile globale e, quando si chiama la funzione, viene utilizzato il suo valore corrente.

3.8 Funzioni su più linee

In QuickBASIC vi sono altri due modi per definire le funzioni: uno è il blocco FUNCTION...END FUNCTION trattato nel Capitolo 5, l'altro è la definizione di funzione su più linee DEF FN...END DEF, qui descritta.

DEF FN...END DEF è una evoluzione dell'istruzione su singola linea DEF FN e permette di definire funzioni più complesse che non possono essere comprese in una singola linea: è dunque molto più potente di una definizione di funzione su singola linea.

Una definizione di funzione su più linee è semplice e chiara: può spesso essere utilizzata in sostituzione delle strutture di subroutine indirette ed i programmi divengono più scorrevoli rispetto a quelli con subroutine. Una funzione su più linee viene definita e designata nel seguente modo:

```
DEF FNspremi$ (stringa$)
.
.
.
END DEF
```

FNspremi$ è il nome della funzione e la parte che segue le lettere FN può essere un qualunque nome legale di variabile. Se la funzione è una funzione numerica, il suo nome deve essere seguito da un identificatore di tipo numerico, mentre se la funzione è stringa il nome deve essere seguito da un identificatore di tipo stringa.

DEF FNspremi$ Definizione di funzione stringa.

DEF FNbis% Definizione di funzione intero.

La linea di apertura della funzione può contenere un elenco di parametri (elenco di nomi di variabili separati da virgole) che nell'esempio consiste in un'unica variabile, stringa$, che è una variabile fittizia. Quando la funzione viene chiamata, viene assegnato a stringa il valore nell'istruzione di chiamata: ad esempio la funzione potrebbe essere chiamata da una istruzione PRINT.

```
PRINT FNspremi$(stringatest$)
```

Il valore assegnato a stringatest$ nel programma verrebbe utilizzato dalla variabile fittizia, stringa$, nella definizione di funzione.

Infine la linea precedente il termine della definizione di funzione assegna al nome della funzione un valore che deve essere passato all'istruzione chiamante nel programma.

```
DEF FNspremi$ (stringa$)
' Elimina gli spazi da una stringa
' e restituisce la stringa compressa

   STATIC spremi$, k, car$
   spremi$ = ""
   FOR k = 1 TO LEN(stringa$)
     car$ = MID$(stringa$, k, 1)
     IF car$ <> " " THEN spremi$ = spremi$ + car$
   NEXT k
   FNspremi$ = spremi$
END DEF
```

Un modo possibile per uscire da una definizione di funzione su più linee è EXIT DEF, utilizzata in particolari condizioni in cui non sia desiderabile l'uscita normale (END DEF); una funzione su più linee deve comunque sempre terminare con END DEF.

La funzione su più linee, essendo essenzialmente una definizione, deve comparire nel programma prima di qualunque istruzione che ne faccia uso, quindi si dovrebbero porre le definizioni di funzione nella parte iniziale del programma.

Il Programma 3.5, SPREMISTRINGHE, illustra una funzione su più linee, FNspremi$, che rimuove tutti gli spazi da una stringa e la Figura 3.5 illustra una esecuzione di prova del programma: per ogni valore di stringatest$ digitato, il programma mostra la stringa compressa con tutti gli spazi rimossi.

```
REM ** SPREMISTRINGHE **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0305.BAS

REM ** Definisce le variabili e la funzione **
DEFINT A-Z
DEF FNspremi$ (stringa$)

' Elimina gli spazi da una stringa
' e restituisce la stringa compressa

   STATIC spremi$, k, car$
   spremi$ = ""
   FOR k = 1 TO LEN(stringa$)
     car$ = MID$(stringa$, k, 1)
     IF car$ <> " " THEN spremi$ = spremi$ + car$
   NEXT k
   FNspremi$ = spremi$
END DEF
```

```
REM ** Programma principale, Test FNspremi$ **
CLS

DO
   INPUT "Stringa, prego"; stringatest$
   PRINT FNspremi$(stringatest$)
   PRINT
   PRINT "Premere Esc per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO

LOOP

END
```

Programma 3.5 SPREMISTRINGHE

```
Stringa, prego? Friuli Venezia Giulia
FriuliVeneziaGiulia

Premere Esc per finire od un altro tasto per continuare
Stringa, prego? T O G L I G L I S P A Z I
TOGLIGLISPAZI

Premere Esc per finire od un altro tasto per continuare
Stringa, prego?  02 7000 18 45
0270001845

Premere Esc per finire od un altro tasto per continuare
```

Figura 3.5 Output del Programma 3.5

FNspremi$ è una funzione stringa di un argomento stringa; l'argomento stringa$ ha significato solamente all'interno della funzione, mentre sarà possibile utilizzare senza conseguenze stringa$ come nome per un'altra variabile in altre parti del programma.

Le variabili spremi$, k e char sono state dichiarate come variabili statiche nel seguente modo:

```
STATIC spremi$, k, car$
```

Le variabili statiche sono locali nella funzione e non hanno alcun significato al di fuori della sua definizione, le si potrebbero quindi utilizzare in altri punti del programma; utilizzando la funzione, i valori di spremi$, k o car$, utilizzati in altre parti del programma fuori della definizione, non vengono modificati.

Si osservi inoltre che spremi$ è utilizzato come parte del nome della funzione ed anche come variabile nella definizione; QuickBASIC non trova difficoltà nel distinguere il nome della funzione, FNspremi$, dalla variabile spremi$ e questo è particolarmente evidente nell'istruzione che assegna il valore di spremi$ come valore della funzione:

```
FNspremi$ = spremi$
```

Una funzione su più linee termina sempre con l'istruzione END DEF.

È possibile scrivere una funzione che rimuova gli spazi da una stringa e trasformi anche le lettere minuscole in maiuscole: quelli che seguono sono due modi per creare una funzione QuickBASIC che assolva questo compito.

- Utilizzare la funzione precedentemente definita FNspremi$ in una definizione di funzione su singola linea.

```
DEF FNspremisu$ (stringa$) = UCASE$(FNspremi$(stringa$))
```

- Definire una funzione su più linee.

```
DEF FNspremisu$ (stringa$)
' Elimina gli spazi da una stringa
' e trasforma le minuscole in maiuscole
  STATIC spremi$, k, car$
  spremi$ = " "
   FOR k = 1 to LEN(stringa$)
   car$ = MID$(stringa$, k, 1)
   IF car$ <> " " THEN spremi$ = spremi$ + car$
NEXT k
FNspremisu$ = UCASE$(spremi$)
END DEF
```

Una funzione su più linee può creare sottostringhe di stringhe lunghe mantenendo solamente alcuni dei caratteri della stringa originale, oppure eliminando alcuni dei caratteri della stringa originale. Seguono due esempi di simili funzioni.

- La funzione FNtienidopo$ costruisce una sottostringa della stringa prima$, utilizzando solamente i caratteri contenuti nella stringa tieni$.

```
DEF FNtienidopo$ (prima$, tieni$)
  STATIC dopo$, k, car$, tieni%
  dopo$ = " "
  FOR k = 1 to LEN(prima$)
   car$ = MID$(prima$, k, 1)
   tieni% = INSTR(tieni$, car$)
   IF tieni% THEN dopo$ = dopo$ + car$
NEXT k
FNtienidopo$ = dopo$
END DEF
```

- La funzione FNeliminadopo$ costruisce una sottostringa della stringa denominata prima$, utilizzando solamente i caratteri non contenuti nella stringa elimina$.

```
DEF FNeliminadopo$ (prima$, elimina$)
STATIC dopo$, k, car$, elimina%

  dopo$ = " "
  FOR k = 1 to LEN(prima$)
   car$ = MID$(prima$, k, 1)
   elimina% = INSTR(elimina$, car$)
   IF elimina% = 0 THEN dopo$ = dopo$ + car$
 NEXT k

FNeliminadopo$ = dopo$

END DEF
```

È opportuno memorizzare queste definizioni di funzione sul dischetto strumenti.

3.9 Strutture di controllo

GW-BASIC ha un insieme di strutture di controllo che vengono utilizzate per prendere decisioni e per eseguire operazioni ripetitive già familiari al lettore:

```
Cicli FOR...NEXT
IF...THEN...ELSE
ON...GOSUB ed ON...GOTO
Cicli WHILE...WEND
```

QuickBASIC comprende tutte queste strutture di controllo e ne ha di altre non disponibili in GW-BASIC, tra le quali:

```
Blocco IF...THEN...ELSE
SELECT CASE
Cicli DO...LOOP
```

Queste nuove strutture sono potenti, flessibili e migliorano la leggibilità del programma; il seguito di questo capitolo descrive in dettaglio ciascuna struttura fornendone degli esempi di uso.

3.9.1 Blocco IF...THEN...ELSE

I blocchi IF...THEN...ELSE aumentano notevolmente l'utilità e la leggibilità degli IF...THEN...ELSE su una sola linea di GW-BASIC. La Tabella 3..2 ne mostra la sintassi ed un semplice esempio.

Tabella 3.2 Sintassi ed esempio di IF...THEN...ELSE

Sintassi	Esempio

IF condizione1 THEN
[bloccoistruzioni-1]
[ELSEIF condizione2 THEN
[bloccoistruzioni-2]]
.
.
.
[ELSE
[bloccoistruzioni-n]]
END IF
	IF numero = 1 THEN
	PRINT "Uno"
	ELSEIF numero = 2 THEN
	PRINT "Due"
	ELSEIF numero = 3 THEN
	PRINT "Tre"
	ELSE
	PRINT "Ne' uno, ne' due, ne' tre"
	END IF

Ecco altri tre esempi:

```
1.  IF numero = INT(numero) THEN          'condizione 1
      PRINT "Intero"                      'bloccoistruzioni-1
    ELSE
      PRINT "Frazionario"                 'bloccoistruzioni-2
    END IF

2.  IF numero < 0 THEN                    'condizione 1
      nzp$ = "negativo"                   'bloccoistruzioni-1
      flag = -1                           'due istruzioni
    ELSEIF numero = 0 THEN                'condizione 2
      nzp$ = "zero"                       'bloccoistruzioni-2
      flag = 0                            'due istruzioni
    ELSE
      nzp$ = "positivo"                   'bloccoistruzioni-3
      flag = 1                            'due istruzioni
    END IF

3.  IF CoV$ = "C" THEN                    'condizione 1
      parola$ = parola$ + consonanti$     'bloccoistruzioni-1
    ELSEIF CoV$ = "V" THEN                'condizione 2
      parola$ = parola$ + vocali$         'bloccoistruzioni-2
    END IF
```

Scrivendo una struttura a blocco IF...THEN...ELSE si tenga presente che:

- una linea che inizia con IF od ELSEIF deve finire con THEN;
- un blocco di istruzioni (bloccoistruzioni) può avere un numero qualsiasi di istruzioni;
- END IF comprende uno spazio, mentre ELSEIF no;
- la struttura IF a blocchi deve essere terminata da una END IF.

Il Programma 3.6, INVENTAPAROLE NUMERO 2, mostra i blocchi IF al lavoro; in questo programma si può digitare la struttura desiderata per la parola ed il numero di parole da inventare.

La struttura è una stringa composta di sole lettere c e v, per specificare consonanti (c) e vocali (v).

```
REM ** INVENTAPAROLE NUMERO 2 **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0306.BAS
' Utilizzato per verificare la FNcasparola$(formaparola$)

REM ** Definisce variabili, poi FUNCTION **
DEFINT A-Z
DEF FNcasparola$ (formaparola$)

' Restituisce una parola a caso nella forma definita dal valore
' di formaparola$. Ad esempio, se formaparola$
' ™ "CVCVC", la parola a caso e' nella forma
' consonante, vocale, consonante, vocale, consonante.

   STATIC parola$, k, CoV$, rc, consonanti$, rv, vocali$
   parola$ = ""
   FOR k = 1 TO LEN(formaparola$)
     CoV$ = UCASE$(MID$(formaparola$, k, 1))
     IF CoV$ = "C" THEN
       rc = INT(21 * RND(1)) + 1
       consonanti$ = MID$("bcdfghjklmnpqrstvwxyz", rc, 1)
       parola$ = parola$ + consonanti$
     ELSEIF CoV$ = "V" THEN
       rv = INT(6 * RND(1)) + 1
       vocali$ = MID$("aeiou", rv, 1)
       parola$ = parola$ + vocali$
     END IF
   NEXT k
   FNcasparola$ = parola$
END DEF

REM ** Programma principale **

DO
   CLS : RANDOMIZE TIMER
   INPUT "Struttura della parola (stringa di C e V)"; struttura$
   INPUT "Quante parole "; numparole
   PRINT
   FOR parola = 1 TO numparole
     PRINT FNcasparola$(struttura$)
   NEXT parola
   PRINT
   PRINT "Premere Esc per finire od un altro tasto per continuare"
```

```
  ta$ = INPUT$(1): IF ta$= CHR$(27) THEN EXIT DO
LOOP

END
```

Programma 3.6 INVENTAPAROLE NUMERO 2

La Figura 3.6 mostra un esempio di esecuzione per la struttura di parola cvcvc.

```
Struttura della parola (stringa di C e V)? CVCVC
Quante parole? 80

suqot    javof    sojej    paron    sajid
dygob    yiyel    fahom    sagob    timyl
hidis    banoh    humab    kibir    jilin
qozap    lugim    godyv    tofat    viziv
budaf    zelul    tivef    vuvih    yuhig
qybiv    cymuh    bered    qejoq    buhog
yypas    nejej    qoxin    zyvef    wuyub
qaruh    buqod    quciv    raseq    qawax
kezet    gayyy    bucom    befac    jynat
nykem    dyduf    tuzof    jugym    kupan
kocyx    tozis    vedey    kahaj    qusuy
vekis    difom    jawog    gyvic    teqip
doleh    xifuh    manom    neger    zalyf
wuxir    yoxiz    paber    kapyy    vitum
pygof    voxeq    fozuw    ropen    byref

Premere Esc per finire od un altro tasto per continuare
```

Figura 3.6 Primo output del Programma 3.6

3.9.2 Struttura SELECT CASE

SELECT CASE è una struttura per decisioni multiple: la Tabella 3.3 ne illustra la sintassi ed un semplice esempio.

Tabella 3.3 Sintassi ed esempio di SELECT CASE

Sintassi	Esempio
SELECT CASE espressione CASE listaespressione1 [bloccoistruzioni-1] CASE listaespressione2 [bloccoistruzioni-2] . . . [CASE ELSE [bloccoistruzioni-n]] END SELECT	SELECT CASE numero CASE 1 PRINT "Uno" CASE 2 PRINT "Due" CASE 3 PRINT "Tre" CASE ELSE PRINT "Non 1,2 o 3" END SELECT

L'espressione che segue SELECT CASE può essere numerica o stringa e l'elenco di espressioni in ogni proposizione CASE deve essere dello stesso tipo (numerico o stringa) dell'espressione che segue SELECT CASE.

La struttura SELECT CASE dell'esempio opera nel modo seguente:

- Se il valore di un numero è 1, 2 o 3, viene stampata la corrispondente stringa ("Uno", "Due" o "Tre").
- Se il valore di un numero non è 1,2 o 3, viene stampata la stringa "Non 1, 2 o 3".

Il contrario dell'esempio illustrato nella Tabella 3.3 è come segue:

```
SELECT CASE NomeNumero$
   CASE "Uno"
    PRINT 1
   CASE "Due"
    PRINT 2
   CASE "Tre"
    PRINT 3
   CASE ELSE
    PRINT "Non Uno, Due o Tre"
END SELECT
```

Una proposizione può specificare un elenco di valori, come nei due seguenti esempi:

```
1.  SELECT CASE numero
      CASE 0, 2, 4, 6, 8
        PRINT "Cifra decimale pari"
        PariODispari = 0
      CASE 1, 3, 5, 7, 9
        PRINT "Cifra decimale dispari"
        PariODispari = 1
      CASE ELSE
```

```
        PRINT "Cifra non decimale"
    END SELECT

2.  SELECT CASE UCASE$(LEFT$(mese$, 3))
      CASE "DIC", "GEN","FEB"
  PRINT "Senti che freddo "
 CASE "MAR", "APR","MAG"
  PRINT "E' primavera"
 CASE "GIU", "LUG","AGO"
  PRINT "Un'estate al mare..."
 CASE "SET", "OTT","NOV"
  PRINT "Come passa il tempo"
 CASE ELSE
  PRINT "Il tempo e' dalla mia parte"
END SELECT
```

Una clausola CASE può dipendere da una espressione relazionale; per indicare ciò, CASE è seguito da IS, nel seguente modo:

```
SELECT CASE numero
 CASE IS < 0
  PRINT "Numero negativo"
 CASE IS 0
  PRINT "Zero"
 CASE IS > 0
  PRINT "Numero positivo"
END SELECT
```

Si è omessa la proposizione CASE ELSE poiché un numero deve essere negativo, zero o nullo; non esistono altre possibilità che possano essere comprese nella proposizione CASE ELSE.

Il Programma 3.7, NUMERO DI GIORNI DI UN MESE, utilizza SELECT CASE per differenziare i mesi che contengono un diverso numero di giorni. SELECT CASE è utilizzata in FNGiorniNelMese%, una funzione che determina il numero di giorni in un mese di un anno normale; il giorno supplementare di febbraio negli anni bisestili è individuato utilizzando FNbis%, una funzione precedentemente già definita in questo capitolo. Si digitino l'anno ed il numero corrispondente ad un qualunque mese dell'anno (1-12) ed il programma stamperà il numero di giorni contenuti in quel mese.

```
REM ** NUMERO DI GIORNI DI UN MESE **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0307.BAS
' Utilizzato per provare FNGiorniNelMese%

REM ** Definisce le variabili e DEF FN **
DEFINT A-Z
' FNbis% restituisce i valori che seguono:
' -1 se l'argomento e' un anno bisestile
'  0 se l'argomento non e' un anno bisestile
```

```
DEF FNbis% (y) = (y MOD 4 = 0 AND y MOD 100 <> 0) OR y MOD 400 = 0

DEF FNGiorniNelMese% (mese)
' FNGiorniNelMese% restituisce il numero di giorni in un
' mese per valore dei mesi da 1 a 12. Restituisce
' 28 per Febbraio (mese = 2) e zero per mesi non validi.
   STATIC giorni
   SELECT CASE mese
     CASE 1, 3, 5, 7, 8, 10, 12
       giorni = 31
     CASE 4, 6, 9, 11
       giorni = 30
     CASE 2
       giorni = 28
     CASE ELSE
       giorni = 0
   END SELECT
   FNGiorniNelMese% = giorni
END DEF

REM ** Programma principale **
CLS
DO
   INPUT "Anno"; anno
   INPUT "Mese (da 1 a 12)"; mese
   numgiorni = FNGiorniNelMese%(mese)
   IF FNbis%(anno) AND mese = 2 THEN numgiorni= numgiorni + 1
   PRINT "Il mese "; mese; "ha"; numgiorni; "giorni."
   PRINT
   PRINT "Premere Esc per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END
```

Programma 3.7 NUMERO DI GIORNI IN UN MESE

La Figura 3.7 illustra un'esecuzione del programma.

```
Anno? 1940
Mese (da 1 a 12)? 2
Il mese 2 ha 29 giorni


Premere Esc per finire od un altro tasto per continuare
Anno? 1941
Mese (da 1 a 12)? 3
```

```
Il mese 3 ha 31 giorni

Premere Esc per finire od un altro tasto per continuare
Anno? 1941
Mese (da 1 a 12)? 4
Il mese 4 ha 30 giorni

Premere Esc per finire od un altro tasto per continuare
```

Figura 3.7 Output del Programma 3.7

Anche il programma 3.8, NUMERO DI GIORNI DI UN MESE-VERSIONE STRINGA, utilizza SELECT CASE, ma per i mesi anziché numeri utilizza stringhe di tre lettere ("GEN", "FEB", "MAR" e così via). La Figura 3.8 illustra un esempio di esecuzione. Il programma fa anche uso di un ciclo DO...LOOP che verrà descritto in seguito.

```
REM ** NUMERO DI GIORNI DI UN MESE, VERSIONE STRINGA **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0308.BAS
' Utilizzato per provare FNGiorniNelMese%

REM ** Definisce variabili e DEF FN **
DEFINT A-Z

DEF FNbis% (y) = (y MOD 4 = 0 AND y MOD 100 <> 0) OR y MOD 400 = 0
' FNbis% restituisce i valori che seguono:
' -1 se l'argomento e' un anno bisestile
'  0 se l'argomento non e' un anno bisestile

DEF FNGiorniNelMese% (mese$)
' FNGiorniNelMese% restituisce il numero di giorni in un
' per valori di mese% = "GEN", "FEB", e cosi' via.
' Restituisce 28 per "FEB" e 0 per valori non validi di mese$.
   STATIC giorni
   SELECT CASE UCASE$(mese$)
     CASE "GEN", "MAR", "MAG", "LUG", "AGO", "OTT", "DIC"
       giorni = 31
     CASE "APR", "GIU", "SET", "NOV"
       giorni = 30
     CASE "FEB"
       giorni = 28
     CASE ELSE
       giorni = 0
   END SELECT
   FNGiorniNelMese% = giorni
END DEF
```

```
REM ** Programma principale**
CLS
DO
   INPUT "Anno "; year
   INPUT "Mese "; mese$
   numgiorni% = FNGiorniNelMese%(LEFT$(mese$, 3))
   IF FNbis%(anno) AND UCASE$(LEFT$(mese$, 3)) = "FEB" THEN
     numgiorni% = 29
   END IF
   PRINT mese$; "ha"; numgiorni%; "giorni."
   PRINT
   PRINT "Premere Esc per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END
```

Programma 3.8 NUMERO DI GIORNI DI UN MESE - VERSIONE STRINGA

```
Anno? 1939
Mese? Febbraio
Febbraio ha 28 giorni.

Premere Esc per finire od un altro tasto per continuare
Anno? 1940
Mese? Febbraio
Febbraio ha 29 giorni.

Premere Esc per finire od un altro tasto per continuare
Anno? 1941
Mese? Marzo
Marzo ha 30 giorni.

Premere Esc per finire od un altro tasto per continuare
Anno? 1941
Mese? Aprile
Aprile ha 31 giorni.
Premere Esc per finire od un altro tasto per continuare
```

Figura 3.8 Output del Programma 3.8

3.9.3 Struttura DO...LOOP

La struttura DO...LOOP permette la ripetizione di un blocco di istruzioni mentre (WHILE) o sino a che non (UNTIL) sia soddisfatta una condizione: è più versatile dei cicli WHILE...WEND e FOR...NEXT per via della sua possibilità di verificare la condizione indifferentemente al principio od al termine del ciclo.

Il ciclo DO...LOOP può assumere tre forme: i programmi precedenti utilizzavano un ciclo infinito che non aveva una condizione di test anche se vi era comunque la possibilità all'interno del ciclo di uscirne (EXIT) premendo il tasto Esc.

```
DO
  INPUT "Anno"; year
  INPUT "Mese "; mese$
  numgiorni% = FNGiorniNelMese%(LEFT$(mese$, 3))
  IF FNbis%(anno) AND UCASE$(LEFT$(mese$, 3)) = "FEB" THEN
    numgiorni% = 29
  END IF
  PRINT mese$; "ha"; numdays; "giorni."
  PRINT
  PRINT "Premere Esc per finire od un altro tasto per continuare"
  ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP
```

Un'altra forma controlla la condizione al principio del ciclo.

```
DO WHILE INKEY$ = ""
   'Questo e' un ciclo vuoto che viene
   'eseguito fino a che non sia premuto un tasto
LOOP
```

Un blocco di istruzioni compreso tra la linea di apertura (DO o WHILE) e la linea di chiusura (LOOP) è costituito da una o più istruzioni BASIC; l'espressione (INKEY$ = "") può essere una qualunque espressione che prende valori vero (non zero) o falso (zero).

Un'altra forma verifica la condizione al termine del ciclo.

```
DO
 INPUT "Transazione"; ammontare
 bilancio = bilancio + ammontare
LOOP WHILE ammontare <> 0
```

La condizione può essere inclusa sia nella proposizione WHILE che nella proposizione UNTIL; ad esempio:

```
DO WHILE INKEY$ = ""
 ' Questo e' un ciclo vuoto che viene
 ' eseguito mentre nessun tasto viene premuto

LOOP
```

```
DO UNTIL INKEY$ <> ""
 ' Questo e' un ciclo vuoto che viene
 ' eseguito fino a quando non venga premuto un tasto
LOOP
```

Entrambe le forme portano allo stesso risultato. Se si desidera eseguire il ciclo almeno una volta, si ponga il test al termine del ciclo (se il test si trovasse al principio del ciclo, la verifica della condizione potrebbe comportare la mancata esecuzione di tutte le istruzioni all'interno del ciclo). Ad esempio, volendo fare un saldo del proprio conto corrente per mezzo di un ciclo, l'esecuzione di almeno un passaggio nel ciclo sarebbe assicurata scrivendo il ciclo nella seguente forma:

```
DO
 INPUT "Transazione"; ammontare
  bilancio = bilancio + ammontare
LOOP WHILE ammontare <> 0
```

Il ciclo verrebbe così eseguito almeno una volta e, se il valore digitato per primo fosse zero, il totale non verrebbe modificato; se si inserisse invece un valore non nullo, questo verrebbe aggiunto al saldo ed il ciclo verrebbe eseguito di nuovo. Si digitino valori negativi per gli assegni staccati e positivi per i depositi. Una forma equivalente è:

```
DO
 INPUT "Transazione"; ammontare
  bilancio = bilancio + ammontare
LOOP UNTIL ammontare = 0
```

L'uscita da un ciclo DO...LOOP può essere eseguita anche con la proposizione facoltativa già illustrata EXIT LOOP, che può comparire in un ciclo DO...LOOP un numero qualunque di volte.

Il Programma 3.9, CONTA I CARATTERI DI UNA STRINGA, fa uso di DO...LOOP con una proposizione WHILE al principio del ciclo nella definizione della funzione, assicurando così che il ciclo non verrà eseguito se la condizione non è soddisfatta; all'interno del ciclo è posta l'istruzione EXIT DO per terminare la ricerca di un carattere.

```
REM ** CONTA I CARATTERI DI UNA STRINGA **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0309.BAS
' Utilizzato per provare FNcontacar%

REM ** Definisce variabili e DEF FN **
DEFINT A-Z
DEF FNcontacar% (stringa$, car$)
' FNcontacar% conta il numero di ricorrenze
' di un singolo carattere (car$) in una stringa (stringa$).
   STATIC conteggio, inizio, trovato
   conteggio = 0
   inizio = 1
   DO WHILE inizio <= LEN(stringa$)
     trovato = INSTR(inizio, stringa$, car$)
```

```
    IF trovato = 0 THEN
      EXIT DO
    ELSE
      conteggio = conteggio + 1
      inizio = trovato + 1
    END IF
  LOOP
  FNcontacar% = conteggio
END DEF

REM ** Programma principale **
CLS
DO
  INPUT "Stringa, prego"; stringatest$
  INPUT "Carattere, prego"; cartest$
  ripetizioni = FNcontacar%(stringatest$, cartest$)
  PRINT "Il carattere si ripete"; ripetizioni; "volte nella stringa."
  PRINT
  PRINT "Premere Esc per finire od un altro tasto per continuare"
  ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END
```

Programma 3.9 CONTA I CARATTERI DI UNA STRINGA

```
Stringa, prego? Questa stringa ha quattro spazi.
Carattere, prego? " "
Il carattere si ripete 4 volte nella stringa.

Premere Esc per finire od un altro tasto per continuare
Stringa, prego? "Questa stringa ha cinque spazi. "
Carattere, prego? " "
Il carattere si ripete 5 volte nella stringa.

Premere Esc per finire od un altro tasto per continuare
Stringa, prego? "1010101010111100101010010 1"
Carattere, prego? "1"
Il carattere si ripete 13 volte nella stringa.

Premere Esc per finire od un altro tasto per continuare
Stringa, prego? "1010101010111100101010010 1"
```

```
Carattere, prego? "1"
Il carattere si ripete 3 volte nella stringa.

Premere Esc per finire od un altro tasto per continuare
```

Figura 3.9 Output del Programma 3.9

Il ciclo DO...LOOP è parte di una definizione di funzione su più linee che tiene il conto del numero di volte che un carattere ricorre in una stringa. Si utilizzi il programma per verificare il funzionamento della funzione di conteggio di caratteri: si digiti una stringa alla prima richiesta ed alla seconda si digiti il carattere di cui si desidera il conteggio, il programma conta quindi quante volte il carattere ricorre nella stringa e stampa il risultato.

Per permettere la terminazione del programma viene utilizzato un ciclo DO...LOOP senza condizione.

3.10 Sommario

In questo capitolo è stata presentata la possibilità di creare un insieme di strumenti di programmazione, di memorizzarli su dischetto ed includerli in seguito in altri programmi.

Sono state trattate le variabili in QuickBASIC, sia locali a particolari porzioni di un programma, che globali all'intero programma, sono stati trattati i tipi di dati e si sono discusse e provate le linee e le etichette di programma di QuickBASIC.

Una trattazione delle funzioni predefinite di QuickBASIC ha portato alle funzioni definite dall'utente: si è provato un insieme assortito di definizioni di funzioni su linea singola e si è trattata in dettaglio ed utilizzata la potente definizione di funzioni su più linee in programmi dimostrativi.

Sono state introdotte e provate le nuove strutture di controllo di QuickBASIC: IF...THEN...ELSE, SELECT CASE e DO...LOOP.

Capitolo 4

Editing

4.1 Menu completo
4.2 Spostamenti in un file
4.3 Inserimento e cancellazione di testo
4.4 Comandi del menu Modifica
4.5 Sommario

Nei Capitoli 1, 2 e 3 l'editing era limitato al controllo del cursore (tasti freccia) ed ai tasti **Backspace** e **Canc** e si è lavorato con il sistema di menu semplificato; in questo capitolo e nel resto del libro si apprenderà l'uso del sistema di menu completo. Questo capitolo presenta i metodi di editing più sofisticati e permette di esercitarsi nel loro uso. Si apprenderà come muovere il cursore e far scorrere lo schermo per raggiungere punti lontani nel programma, si proverà ad inserire e cancellare piccole porzioni di testo, verrà trattato l'uso dei comandi del menu Modifica del sistema di menu completo per manipolare grandi quantità di testo e si utilizzeranno inoltre i comandi Taglia, Annulla, Incolla, Merge e Copia. QuickBASIC esegue automaticamente alcune funzioni, tra cui:

- conversione delle parole chiave QuickBASIC in maiuscolo, se digitate in minuscolo;
- inserimento degli spazi prima e dopo tutti gli operatori, se già non esistevano;
- controllo della sintassi, memoria ed errori di definizioni ripetuti;
- esecuzione di modifiche di minore entità, come l'inserimento di spazi, in modo che il codice sorgente QuickBASIC si presenti in un formato standard.

4.1 Menu completo

Si accede al sistema di menu completo attraverso il menu Opzioni, evidenziando il comando Menu completi e premendo **Invio**: si ritorna così alla Finestra di visualizzazione ed

all'editor di QuickBASIC. Come primo esercizio con il sistema di menu completo, si digiti un nuovo programma e si acceda al menu File, che ora contiene un elenco di comandi più lungo. La Tabella 4.1 elenca le voci per i due tipi di menu.

Si osservi che tutte le voci del menu File nel sistema semplificato compaiono anche in quello completo, ma vi sono in più molte nuove voci.

Tabella 4.1 Menu File per sistemi di menu semplificato e completo

Menu File semplificato	Menu file completo
Nuovo programma	Nuovo programma
Apri programma	Apri programma
Salva con un nome	Merge
Stampa	Salva
	Salva con nome
Uscita	Salva tutto
	Crea un file
	Carica un file
	Scarica un file
	Stampa
	Shell DOS
	Uscita

Per introdurre un nuovo programma, si evidenzi nuovamente Nuovo programma e si prema **Invio**; si è già utilizzato in precedenza questo comando, che mette a disposizione una Finestra di visualizzazione vuota in cui introdurre il Programma 4.1, NOMI DEI FILE DI UTILITÀ. Si digiti il programma così come è, anche se si individuano degli errori.

Questo programma è costituito da sole istruzioni REM e non ha quindi istruzioni eseguibili: il suo nome è UQB0001.BAS per farlo apparire sempre per primo nell'elenco dei programmi visualizzati nel riquadro di comunicazione Apri programma. Con la scrittura di altri programmi, questo programma di istruzioni REM diverrà sempre più utile; quando si desidererà revisionare od utilizzare un programma si potrà ricorrere al file UQB0001.BAS per individuare il programma da richiamare. Lo si utilizzi ora per saggiare le tecniche di editing che seguono.

```
REM ** NOMI DEI PROGRAMMI DI UTILITA' **
' Programma REM - nomi dei filed e descrizioni
' Capitolo 4
' Micosoft QuickBASIC  4.5 File: UQB0001.BAS

REM ** Capitolo 1 **
' UQB0106.BAS Numerologia con grafica
```

```
REM ** Capitolo 2 **
' UQB0201.BAS Lancio monete

REM ** Capitolo 3 **
' UQB0301.BAS Dollari e cent
' UQB0302.BAS Minimo e massimo
' UQB0303.BAS Anno bisestile
' UQB0304.BAS Inventaparole numero 1
' UQB0305.BAS Spremistringhe
' UQB0306.BAS Inventaparole numero 2
' UQB0307.BAS Numero di giorni di un mese
' UQB0308.BAS Numero di giorni di un mese, versione stringa
```

Programma 4.1 NOMI DEI FILE DI UTILITÀ

4.2 Spostamenti in un file

Mentre ci si trova nella Finestra di visualizzazione è possibile introdurre del testo, spostare il cursore lungo il file ed inserire, sostituire o cancellare testo; ogni operazione di editing richiede tre passi:

- portare il cursore nella posizione desiderata;
- selezionare il testo che deve essere modificato;
- inserire, sostituire o cancellare il testo.

La Tabella 4.2 descrive alcuni metodi per spostare il cursore nello schermo da estremo ad estremo, di linea in linea, di parola in parola e di carattere in carattere; descrive anche come far scorrere lo schermo.

Tabella 4.2 Movimenti del cursore e scorrimento

Movimento del cursore	Tasto da premere
A inizio file	**Ctrl+Iniz**
Alla fine del File	**Ctrl+Fine**
Una linea in alto	**Freccia Su**
Una linea in basso	**Freccia Giù**
All'inizio della linea	**Inizio**
Alla fine della linea	**Fine**
A sinistra di una parola	**Ctrl+Freccia Sinistra**
A destra di una parola	**Ctrl+Freccia Destra**

Movimento del cursore	Tasto da premere
A sinistra di un carattere	**Freccia Sinistra**
A destra di un carattere	**Freccia Destra**
Scorrimento	
A sinistra di una finestra	**Ctrl+PgInd**
A destra di una finestra	**Ctrl+PgAv**
Una finestra verso l'alto	**PgInd**
Una finestra verso il basso	**PgAv**
Una linea verso l'alto	**Ctrl+Freccia Giù**
Una linea verso il basso	**Ctrl+Freccia Su**

Per alcuni movimenti del cursore è necessario premere un tasto mentre se ne tiene premuto un altro e tale combinazione è indicata da un segno più tra i due tasti. Ad esempio **Ctrl+Fine** significa che si deve premere il tasto **Fine** mentre si tiene premuto il tasto **Ctrl.**

Si memorizzi il programma con il nome UQB0001.BAS: lo si utilizzerà per esercitarsi a spostare il cursore ed a correggere il testo, dapprima solamente con piccole modifiche, in seguito cancellando, spostando ed inserendo blocchi di testo. Alla conclusione del capitolo si apprenderà come inserire un file in un altro.

4.2.1 Spostamento al principio ed al termine di un file

Digitando il Programma 4.1 si è probabilmente notato che le prime linee del programma sono scomparse scorrendo oltre il margine superiore dello schermo ed il cursore, dopo aver digitato l'ultima linea, è rimasto lampeggiante al termine del programma.

```
' Micosoft QuickBASIC 4.5   File: UQB0001.BAS


REM ** Capitolo 1 **
 .
 .
 .
' UQB0307.BAS  Numero di giorni di un mese
' UQB0308.BAS  Numero di giorni di un mese, versione stringa
_
```

^ cursore

Si prema **Ctrl+Iniz** ed il cursore si sposterà all'estremo superiore sinistro, sotto la R della prima istruzione REM; in altre parole, si sposta al principio del file.

```
REM ** NOMI DEI PROGRAMMI DI UTILITA' **
^cursore
.
.
.
' UQB0305.BAS Spremistringhe
' UQB0306.BAS Inventaparole numero 2
```

Per spostare il cursore al termine del file si prema **Ctrl+Fine** e lo schermo mostrerà la parte finale del programma, con il cursore posto sull'ultima linea.

Ctrl+Fine *sposta il cursore al termine del file,* **Ctrl+Iniz** *lo sposta al principio del file.*

4.2.2 Spostamento una linea verso l'alto o verso il basso

Si prema, se non lo si è già fatto, **Ctrl+Iniz** per portare il cursore all'inizio del file; il cursore si trova ora sulla prima linea, sopra la R dell'istruzione REM: si prema il tasto **Freccia Giù** ed il cursore si sposterà di una linea verso il basso, rimanendo sulla stessa colonna.

```
REM ** NOMI DEI PROGRAMMI DI UTILITA' **
'_Programma REM - nomi dei filed e descrizioni
^cursore
```

Si prema alcune volte il tasto **Freccia Giù**, ed il cursore si sposterà ogni volta di una linea verso il basso; si prema quindi il tasto **Freccia Su** alcune volte, sino a quando il cursore non sia tornato al principio del file.

Il tasto **Freccia Giù** *sposta il cursore verso il basso di una linea, ii tasto* **Freccia Su** *lo sposta verso l'alto di una linea.*

4.2.3 Spostamento al principio ed alla fine di una linea

Si sposti il cursore al principio della terza linea sullo schermo.

```
_'Capitolo 4
```

^cursore

Si prema quindi il tasto **Fine** ed il cursore si sposterà al termine della linea.

```
'Capitolo 4_
```

^cursore

Si prema il tasto **Iniz** per spostare di nuovo il cursore all'inizio della linea.

Il tasto **Fine** *sposta il cursore al termine di una linea, il tasto* **Iniz** *al suo principio.*

4.2.4 Spostamento a sinistra ed a destra di una parola

Si sposti il cursore dal principio della terza linea al principio della seconda premendo il tasto **Freccia Su**.

```
_' Programma REM - nomi dei filed e descrizioni
```

^cursore

Si prema **Ctrl+Freccia Destra** ed il cursore si sposterà di una parola sulla destra, quindi si prema **Ctrl+Freccia Destra** sino a che il cursore non si trovi sulla parola errata filed. Si utilizzi il tasto Freccia Destra per spostare il cursore alla lettera d della parola e si prema **Canc** per cancellare la lettera: si corregge così uno degli errori nel programma originale.

```
' Programma REM - nomi dei file e descrizioni
```

Si prema **Ctrl+Freccia Sinistra** ed il cursore si sposterà di una parola sulla sinistra: utilizzando questa combinazione di tasti si riporti il cursore alla prima lettera della seconda linea.

Ctrl+Freccia Destra *sposta il cursore di una parola sulla destra, mentre* **Ctrl+Freccia Sinistra** *lo sposta di una parola sulla sinistra.*

Che cosa accade se il cursore si trova già alla prima parola della seconda linea quando si preme **Ctrl+Freccia Sinistra**? Il cursore si sposta forse sull'apostrofo (') all'inizio della linea? Il cursore si sposta alla prima linea? Scompare dall'estremo sinistro dello schermo? Non accade nulla poiché non può spostarsi più verso sinistra nella linea? Si verifichi

provando (il cursore si sposta all'ultima parola della prima linea: si osservi che l'apostrofo e gli asterischi non sono stati considerati delle parole).

4.2.5 Spostamento a destra ed a sinistra di un carattere

Ora che si è appreso come spostarsi nel file a passi più o meno lunghi, si porti il cursore al principio della quarta linea, nella quale è scritta scorrettamente la parola Microsoft e si prema il tasto **Freccia Destra** una volta: il cursore si sposterà di un carattere sulla destra ogni volta che lo si preme, si prema dunque il tasto **Freccia Destra** sino a portarsi sotto la prima o di Micosoft.

```
' Micosoft QuickBASIC 4.5  File: UQB0001.BAS
```

^cursore

Si prema **r**. La lettera viene inserita nella posizione del cursore e tutti i caratteri alla sua destra vengono spostati verso destra, si è così corretto il secondo errore del programma originale. Il cursore si sposta di un carattere sulla sinistra ogni volta che si preme il tasto **Freccia Sinistra**; lo si utilizzi per riportare il cursore al principio della quarta linea.

Il tasto **Freccia Destra** *sposta il cursore sulla destra di un carattere, il tasto* **Freccia Sinistra** *lo sposta a sinistra di un carattere.*

La linea fisica di QuickBASIC è lunga 255 caratteri: che cosa può accadere se si preme il tasto **Freccia Destra** quando il cursore si trova al termine della linea?

Che cosa accade se si trova al principio della linea e si preme il tasto **Freccia Sinistra**?

Si provino entrambe le operazioni (il cursore non si sposterà oltre il termine della linea fisica). Se si sono seguite tutte le direttive, si dovrebbe ora essere in grado di spostare il cursore ovunque all'interno del file; si sposti il cursore nel file per individuare eventuali altri errori e se li si trovano li si correggano.

4.2.6 Scorrimento dello schermo verso l'alto o verso il basso di una finestra

Le funzioni di scorrimento muovono le linee nella finestra come se si stesse srotolando un rotolo; si posizioni il cursore sotto la prima n nella parola nomi nella seconda linea del Programma 4.1.

```
' Programma REM - nomi dei file e descrizioni
                    ^cursore
```

Si prema **PgAv** e lo schermo scorrerà alla seconda finestra del programma. Il cursore si trova ora alla seconda linea della seconda finestra del programma e cioè nella stessa posizione relativa (seconda linea, diciannovesima colonna), è cambiata solamente la finestra visualizzata.

```
' UQB0308.BAS Numero di giorni di un mese, versione stringa
                    ^cursore
```

Si prema ora **PgInd** e lo schermo mostrerà nuovamente la prima finestra del programma.

PgAv *fa scorrere verso il basso di una finestra,* **PgInd** *verso l'alto.*

4.2.7 Scorrimento di una linea verso l'alto o verso il basso

Si sposti ora il cursore verso il basso fino alla lettera A della parola QuickBASIC nella quarta linea del programma, si prema quindi **Ctrl+Freccia Giù** ed il cursore resterà dove era (sotto la lettera A), ma il testo nello schermo scorrerà verso l'alto di una linea. La linea all'estremo superiore scomparirà dallo schermo:

```
' Programma REM - nomi dei file e descrizioni
' Capitolo 4
' Microsoft QuickBASIC 4.5 File: UQB0001.BAS
.                   ^cursore
.
.
```

All'estremo inferiore appare una nuova linea; si prema ora **Ctrl+Freccia Su** ed il testo scorrerà verso il basso di una linea, la linea all'estremo inferiore scomparirà e comparirà una nuova linea all'estremo superiore riportando lo schermo dove già era quando si è premuto **Ctrl+Freccia Giù**.

Ctrl+Freccia Giù *fa scorrere la finestra di visualizzazione verso il basso di una linea,* **Ctrl+Freccia Su** *la fa scorrere verso l'alto.*

4.2.8 Scorrimento di una finestra verso destra o verso sinistra

È possibile anche far scorrere il testo orizzontalmente oltre che in verticale: far scorrere lo schermo verso destra o verso sinistra è come arrotolare e srotolare un rotolo lateralmente.

Si prema **Ctrl+PgAv** con il cursore sotto la lettera A nella quarta linea del Programma 4.1: verrà visualizzata una finestra vuota dal momento che lo schermo si è spostato di una finestra verso destra e, poiché le linee del programma non si estendono oltre la finestra originale, nulla compare in questa.

Si prema **Ctrl+PgInd** e lo schermo scorrerà di una finestra verso sinistra, nuovamente nella posizione iniziale.

Ctrl+PgAv *fa scorrere lo schermo di una finestra verso destra,* **Ctrl+PgInd** *verso sinistra.*

Con uno o più movimenti del cursore o scorrimenti dello schermo è possibile posizionare il cursore ovunque all'interno del file.

4.3 Inserimento e cancellazione di testo

L'editor di QuickBASIC permette di operare in due modi: modo inserimento e modo sostituzione; il tasto **Ins** permette di selezionare alternativamente i due modi.

Quando si accede all'editor è inizialmente attivo il modo inserimento ed il cursore appare come un carattere di sottolineatura (_).

In modo inserimento i caratteri digitati vengono inseriti nella posizione in cui si trova il cursore; il carattere sopra al cursore e quelli che lo seguono vengono spostati di una posizione a destra per fare spazio al nuovo carattere.

Quando è attivo il modo inserimento, premendo il tasto **Ins** si dispone l'editor a funzionare in modo sostituzione ed il cursore si trasforma in un quadratino pieno lampeggiante.

Quando l'editor è in modo sostituzione il carattere alla posizione del cursore viene rimpiazzato dal carattere digitato e gli altri nella linea rimangono immutati.

Il cursore si presenta come un quadratino pieno quando è attivo il modo sostituzione, mentre si presenta come un carattere di sottolineatura quando è attivo il modo sostituzione: premendo il tasto **Ins** *si disattiva il modo corrente e si attiva l'altro.*

La Tabella 4.3 illustra dei metodi per inserire e cancellare un carattere, una linea o parte di una linea di testo ed illustra anche come iniziare una nuova linea di testo alla posizione corrente del cursore.

Tabella 4.3 Inserimento e cancellazione di testo

Per iniziare una nuova linea	**Tasto da premere**
Alla corrente posizione del cursore	**Invio**
Per inserire	
Una linea sopra quella corrente	**Ctrl+N od Iniz, Ctrl+N**
Una linea sotto quella corrente	**Fine+Invio**
Per cancellare	
L'intera linea corrente	**Ctrl+Y**
Sino al termine della linea corrente	**Ctrl+Q, Y**
Il carattere alla sinistra del cursore	**Backspace**
Il carattere sotto al cursore	**Canc**

4.3.1 Inizio di una nuova linea

Nell'introdurre i programmi precedenti, si sono iniziate nuove linee premendo il tasto **Invio** dopo aver digitato l'intera linea, tuttavia il tasto **Invio** può essere utilizzato per iniziare una nuova linea da qualunque posizione. Si supponga di aver digitato la seguente linea di programma:

```
' Elimina gli spazi da una stringa e restituisce la stringa compressa
```

Per rendere il testo più leggibile è possibile suddividere la linea in due parti, con la seconda di queste spostata su una nuova linea; per fare ciò si sposti il cursore alla lettera e e si prema **Invio**: viene fatto spazio per una nuova linea nella quale viene riportata la seconda parte della linea.

```
'Elimina gli spazi da una stringa
e restituisce la stringa compressa
^cursore
```

Si digiti un'apostrofo ('') seguito da uno spazio per rendere la linea un commento anziché un'istruzione.

4.3.2. Inserimento di testo

Quando l'editor è in modo inserimento, è possibile inserire del testo:

- si porti il cursore nella posizione alla quale si desidera inserire;

- si digiti il testo da inserire.

Questo metodo è l'ideale quando si desidera inserire piccole quantità di testo, come pochi caratteri od una o due parole: non appena si digita il testo da inserire tutti i caratteri sotto e sulla destra del cursore verranno spostati a destra di una posizione per ogni carattere inserito. L'editor riformatta la linea dopo l'inserimento.

Il Programma 3.5, SPREMISTRINGHE (riportato qui di seguito per comodità), verrà utilizzato per provare brevi inserzioni ed in seguito per inserire maggiori quantità di testo. Si consiglia di formattare un nuovo dischetto per i programmi che si utilizzeranno nel seguito di questo capitolo per fare esercizio con le proprie tecniche di editing: si copi il Programma 3.5 (UQB0305.BAS) su questo dischetto di esercizi assicurandosi di conservare il programma originale.

```
REM ** SPREMISTRINGHE **
' Capitolo 3
' Microsoft QuickBASIC 4.5  File: UQB0305.BAS

REM ** Definisce le variabili e la funzione **

DEFINT A-Z
DEF FNspremi$ (stringa$)
' Elimina gli spazi da una stringa
' e restituisce la stringa compressa
   STATIC spremi$, k, car$
   spremi$ = ""
   FOR k = 1 TO LEN(stringa$)
     car$ = MID$(stringa$, k, 1)
     IF car$ <> " " THEN spremi$ = spremi$ + car$
   NEXT k
   FNspremi$ = spremi$
END DEF

REM ** Programma principale, Test FNspremi$ **
CLS
DO
   INPUT "Stringa, prego"; stringatest$
   PRINT FNspremi$(stringatest$)
   PRINT
   PRINT "Premere ESC per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP
END
```

Programma 4.2 SPREMISTRINGHE (copia)

Si modificherà il Programma 3.5 attraverso i seguenti passi:

- modifica del titolo in SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA;

- modifica del numero del capitolo in Capitolo 4 anziché Capitolo 3;
- modifica del nome del file da UQB0305.BAS a UQB0403.BAS;
- inserimento di un blocco di linee di programma intestato Variabili utilizzate.

Quando si è preparato il dischetto degli esercizi, si acceda al programma e si porti il cursore al primo asterisco che segue la parola SPREMISTRINGHE nella prima linea.

```
REM ** SPREMISTRINGHE **
                       ^cursore
```

Si prema **Ins** per attivare il modo inserimento nel caso non fosse già attivo (il cursore dovrebbe presentarsi come un carattere di sottolineatura), si digiti **CON DESCRIZIONE DELLE VARIABILI STRINGA** e si prema la **Barra spaziatrice**.

```
REM ** SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA **
                                                               ^cursore
```

Ora, sempre in modo inserimento, si sposti il cursore al numero 3 nella seconda linea del programma.

```
REM ** SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA **
' Capitolo 3
          ^cursore
```

Si digiti **4** e lo schermo presenterà:

```
REM ** SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA **
' Capitolo 43
           ^cursore
```

Si prema il tasto **Canc** per rimuovere il numero 3 ed il capitolo sarà così modificato in 4; si sposti quindi il cursore sul 3 nel nome del file nella terza linea del programma.

```
REM ** SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA **
' Capitolo 4
' Microsoft QuickBASIC 4.5  File: UQB0305.BAS
                                   ^cursore
```

Si digiti **403** e lo schermo presenterà:

```
REM ** SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA **
' Capitolo 4
' Microsoft QuickBASIC 4.5  File: UQB0403305.BAS
                                        ^cursore
```

Si prema il tasto **Canc** tre volte in modo da cancellare i precedenti numeri del programma (305), il nome di file è ora UBQ0403.BAS.

Il programma è ora pronto per l'inserimento di una porzione di testo più consistente: lo si tenga in memoria. Se si devono eseguire inserimenti di porzioni di testo lunghe (alcune linee, un paragrafo o più) è conveniente utilizzare i metodi di inserimento illustrati nella Tabella 4.3.

Per inserire una linea sopra ad un'altra si prema il tasto **Ins** per attivare il modo inserimento dell'editor (se non era già attivo), si sposti il cursore al principio della linea che dovrà seguire l'inserzione e si prema **Ctrl+N**, aprendo così una linea vuota nella quale si troverà a lampeggiare il cursore. È in possibile quindi digitare il testo da inserire nella linea vuota.

Si supponga ad esempio di voler inserire una sezione che descriva le variabili utilizzate nel programma e porla tra il blocco DEF FN ed il programma principale, il blocco Test FNspremi$: si porti il cursore al principio dell'istruzione REM del programma principale.

```
END DEF

REM ** Programma principale, Test FNspremi$ **
^cursore
```

Si prema **Ctrl+N** per aprire una nuova linea sopra l'istruzione REM; questa viene spostata verso il basso di una linea ed il cursore si trova, pronto per l'inserzione, al principio della nuova linea vuota.

```
END DEF
^cursore
REM ** Programma principale, Test FNspremi$ **
```

Si inseriscano le seguenti linee:

```
REM ** Variabili utilizzate **
' spremi$      stringa con spazi rimossi
' car$         un carattere della stringa data
' stringatest$ stringa data per provare FNspremi$
' k            contatore dei caratteri nel ciclo FOR-NEXT
```

Si è così completato il Programma 4.3, SPREMISTRINGHE CON LA DESCRIZIONE DELLE VARIABILI STRINGA; lo si memorizzi nel proprio dischetto di esercizi (utilizzando il suo nuovo nome) per poterlo utilizzare in seguito, ma lo si lasci in memoria per esercitarsi nella prossima sezione a cancellare del testo.

```
REM ** SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA **
' Capitolo 4
' Microsoft QuickBASIC 4.5  File: UQB0403.BAS
REM ** Definisce le variabili e la funzione **
DEFINT A-Z
DEF FNspremi$ (stringa$)
' Elimina gli spazi da una stringa
' e restituisce la stringa compressa
   STATIC spremi$, k, car$
   spremi$ = ""
   FOR k = 1 TO LEN(stringa$)
     car$ = MID$(stringa$, k, 1)
     IF car$ <> " " THEN spremi$ = spremi$ + car$
   NEXT k
   FNspremi$ = spremi$
END DEF

REM ** Variabili utilizzate **
' spremi$      stringa con spazi rimossi
' car$         un carattere della stringa data
' stringatest$ stringa data per provare FNspremi$
' k            contatore dei caratteri nel ciclo FOR-NEXT

REM ** Programma principale, Test FNspremi$ **
CLS
DO
   INPUT "Stringa, prego"; stringatest$
   PRINT FNspremi$(stringatest$)
   PRINT
   PRINT "Premere ESC per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END
```

Programma 4.3 SPREMISTRINGHE CON LA DESCRIZIONE DELLE VARIABILI STRINGA

In questo metodo si è utilizzato **Ctrl+N** per aprire una linea sopra a quella nella quale si trovava il cursore, ma, per aprire una linea allo scopo di inserire testo, si sarebbe potuta utilizzare un'altra combinazione di tasti tra quelle elencate nella Tabella 4.3. Premendo **Fine+Invio** si apre una linea sotto a quella in cui si trova il cursore.

Si prema **Ctrl+N** *(o* **Iniz, Ctrl+N***) per inserire una linea sopra alla posizione del cursore, si prema* **Fine+Invio** *per inserirla sotto la posizione del cursore.*

4.3.3 Cancellazione di testo

Un metodo per cancellare caratteri singoli è quello di portare il cursore sul carattere da cancellare e premere il tasto **Canc**, un altro è quello di portare il cursore sul carattere immediatamente a destra di quello che si desidera cancellare e premere il tasto **Backspace**. Tenendo il Programma 4.3 in memoria, si porti il cursore sull'apostrofo alla sinistra della k nel blocco Variabili utilizzate.

```
REM ** Variabili utilizzate **
' spremi$       stringa con spazi rimossi
' car$          un carattere della stringa data
' stringatest$ stringa data per provare FNspremi$
' k             contatore dei caratteri nel ciclo FOR-NEXT
```

^cursore

Si prema tre volte il tasto **Canc**, eliminando ogni volta un carattere o spazio: tutti i caratteri sulla destra del cursore vengono spostati di una posizione verso sinistra; il risultato è:

```
contatore dei caratteri nel ciclo FOR-NEXT
```

Premendo il tasto **Canc** *si cancella il carattere su cui è posizionato il cursore.*

Si porti il cursore sulla destra della lettera t nella parola FOR-NEXT:

```
contatore dei caratteri nel ciclo FOR-NEXT_
```

^cursore

Ogni volta che si preme il tasto **Backspace** verrà cancellato il carattere alla sinistra del cursore: lo si prema sino a che non venga cancellata tutta la parola FOR-NEXT. Il cursore si trova sullo spazio che segue la parola ciclo.

```
contatore dei caratteri nel ciclo_
```

^cursore

Premendo il tasto **Backspace** *si cancellano i caratteri alla sinistra del cursore e si sposta il cursore di una posizione sulla sinistra.*

Si porti ora il cursore al principio della linea premendo il tasto **Iniz**.

```
_            contatore dei caratteri nel ciclo
```

^cursore

Si prema **Ctrl+Q**. Si osservino i caratteri ^Q che compaiono sulla destra nella linea inferiore dello schermo:

```
<MAIUSC+F1=Guida> <F6=Fin.> <F2=SUB> <F5=Esegui> <F8=Passo> ^Q 00023:001
```

^segnala la pressione di **Ctrl+Q**

Si prema ora Y e verrà cancellata l'intera linea.

Premendo **Ctrl+Q** *ed* **Y** *si elimina tutto dalla posizione del cursore al termine della linea ed il cursore si porta al principio di quella nella quale era stato cancellato il testo.*

Si sposti il cursore verso l'alto di una linea.

```
REM ** Variabili utilizzate **
' spremi$       stringa con spazi rimossi
' car$          un carattere della stringa data
_ stringatest$ stringa data per provare FNspremi$
```

^cursore

Si prema **Ctrl+Y** e l'intera linea scomparirà.

Premendo **Ctrl+Y** *si cancella l'intera linea nella quale si trova il cursore.*

Si eliminino tutte le linee del blocco Variabili nel Programma 4.3 utilizzando uno qualsiasi di questi metodi. Eccetto che per il nome di file ed il numero del capitolo nel blocco di apertura, il programma è nuovamente identico a quello di partenza.

4.4. Comandi del menu Modifica

Dal momento che ora si sta operando con il sistema di menu completo, si mandi a memoria la Tabella 4.4 che elenca i comandi del menu Modifica dei sistemi di menu semplificato e completo. Per copiare od inserire grandi quantità di testo è possibile utilizzare i comandi di uno qualunque dei due sistemi, ma utilizzando il sistema completo, con il comando Merge, è possibile inserire interi programmi.

Tabella 4.4 Menu Modifica semplificato e completo

Menu Modifica semplificato	Menu Modifica completo
Taglia	Annulla
Copia	Taglia
Incolla	Copia
	Incolla
	Cancella
	Nuova SUBroutine
	Nuova FUNCTION

4.4.1 Taglia, Annulla, Copia ed Incolla

Combinando i comandi Taglia ed Incolla del menu Modifica è possibile cancellare una parte di testo ed inserirla in un'altra posizione.

4.4.1.1 Selezione del testo

Prima di poter cancellare od inserire del testo con il menu Modifica è necessario selezionare il blocco di testo da trattare: mentre si è nell'editor si porti il cursore al principio della parte di testo che deve essere selezionata, quindi, tenendo premuto il tasto **Shift**, si evidenzi il testo sino al termine del blocco utilizzando uno dei tasti per il controllo del cursore (**Freccia Sinistra**, **Freccia Destra**, **Ctrl+Freccia Destra**, **PgInd** e gli altri).

Per sperimentare tutto ciò si richiami nuovamente il Programma 4.3: la Figura 4.1 illustra la parte di programma che compare nella Finestra di visualizzazione.

```
REM ** SPREMISTRINGHE CON DESCRIZIONE DELLE VARIABILI STRINGA **
 ' Capitolo 4
 ' Microsoft QuickBASIC 4.5  File: UQB0403.BAS

 REM ** Definisce le variabili e la funzione **
 DEFINT A-Z
 DEF FNspremi$ (stringa$)
 ' Elimina gli spazi da una stringa
 ' e restituisce la stringa compressa
```

```
    STATIC spremi$, k, car$
    spremi$ = ""
    FOR k = 1 TO LEN(stringa$)
      car$ = MID$(stringa$, k, 1)
      IF car$ <> " " THEN spremi$ = spremi$ + car$
    NEXT k
    FNspremi$ = spremi$
  END DEF
```

Figura 4.1 Porzione visualizzata nella finestra del Programma 4.3

Volendo selezionare per la correzione il blocco Variabili utilizzate del Programma 4.3, si porti il cursore sulla R dell'istruzione REM al principio del blocco.

```
REM ** Variabili utilizzate **
^cursore
' spremi$      stringa con spazi rimossi
  .
  .
  .
```

Si prema **Shift+Freccia Giù**: l'istruzione verrà visualizzata in modo reverse (qui riportato in italico) per indicare che è stata selezionata ed il cursore si sarà spostato verso il basso di una linea; si prema nuovamente **Shift+Freccia Giù**, anche la seconda linea del blocco verrà visualizzata in reverse ed il cursore si sarà portato alla terza linea.

```
REM ** Variabili utilizzate **
' spremi$      stringa con spazi rimossi
' car$         un carattere della stringa data
^cursore
  .
  .
  .
```

Si prema **Shift+Freccia Giù** altre tre volte in modo da selezionare in tutto cinque linee, l'intero blocco Variabili utilizzate.

```
REM ** Variabili utilizzate **
' spremi$       stringa con spazi rimossi
' car$          un carattere della stringa data
' stringatest$ stringa data per provare FNspremi$
' k             contatore dei caratteri nel ciclo FOR-NEXT
```

–
^cursore

.

.

.

Si lasci questo blocco evidenziato.

4.4.1.2 Taglia

Per cancellare il testo appena selezionato è necessario accedere al menu Modifica ed evidenziare il comando Taglia, come illustrato nella Figura 4.2.

Per portare a termine la cancellazione si prema il tasto **Invio**: il blocco Variabili utilizzate verrà cancellato, ma, benché il blocco non sia più visibile nel programma, è ancora ricuperabile: è stato memorizzato in una area della memoria detta ritaglio e può essere recuperato, come si vedrà in seguito.

Se si è certi di voler eliminare il blocco di testo è possibile selezionare la parte e premere il tasto **Canc**, ma non sarà più possibile recuperarlo.

Un altro metodo per cancellare del testo consiste nell'attivare il comando Cancella del menu Modifica, ma anche il testo cancellato con il comando Cancella è irrecuperabile. In seguito alla cancellazione del blocco Variabili utilizzate, il programma è nuovamente identico al Programma 4.2, ad eccezione del blocco iniziale, e non è dunque necessario memorizzarlo poiché se ne dispone già di una copia su dischetto.

4.4.1.3 Annulla

Il comando Annulla non è disponibile nel sistema di menu semplificato, ma lo è in quello completo, come illustrato nella Tabella 4.4; rimane disabilitato sino a che non venga modificata la linea corrente e ne ripristina lo stato originale a condizione che il cursore si trovi ancora sulla linea, mentre non ha effetto se si è spostato dalla linea modificata. Si richiami il Programma 4.3 per comprendere come funziona il comando Annulla.

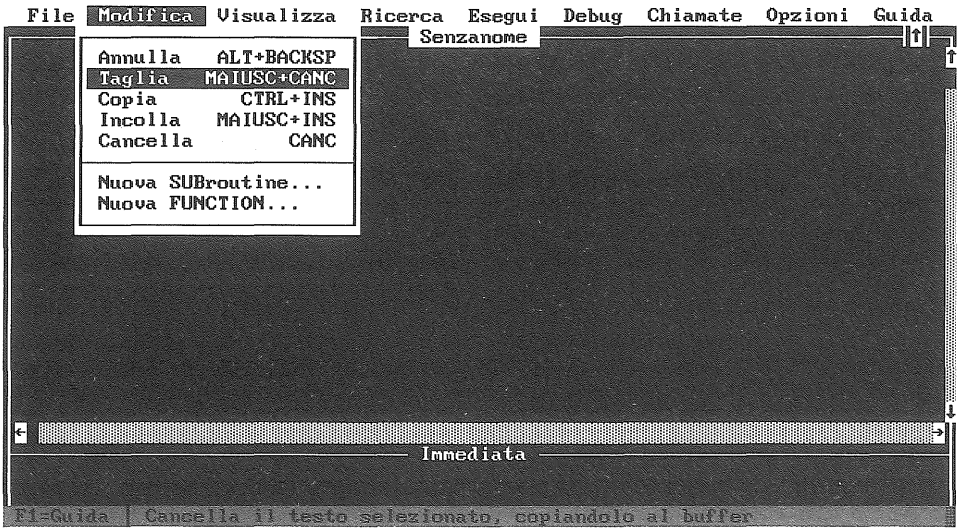


Figura 4.2 Menu Modifica con il comando Taglia evidenziato

Si supponga di voler eliminare le parole utilizzate dalla linea REM del blocco Variabili utilizzate: si selezioni la parola posizionando il cursore sotto la u e premendo **Shift+Freccia Destra** sino a che non sia evidenziata ogni lettera della parola.

```
REM ** Variabili utilizzate _**
```

Si prema **Alt+M** per accedere al menu Modifica e si cancelli la parola utilizzate evidenziando il comando Taglia e premendo **Invio**.

```
REM ** Variabili _**
```

Poiché si è modificata la linea corrente ed il cursore si trova ancora sulla stessa, è possibile ripristinare lo stato originale selezionando il comando Annulla del menu Modifica e premendo **Invio**.

```
REM ** Variabili utilizzate **
```

^cursore

Il cursore si porta al principio della linea ripristinata.

4.4.1.4 Taglia ed Incolla

Quando si elimina del testo con il comando Taglia, il testo cancellato viene posto nel ritaglio ed è ancora accessibile; è perciò possibile inserire il testo cancellato in qualunque altro punto del programma in memoria: il comando Incolla inserisce il testo del ritaglio alla posizione corrente del cursore. I comandi Taglia ed Incolla vengono utilizzati in combinazione per spostare un blocco di testo da un punto all'altro del programma.

Si supponga di voler modificare l'ordine dei blocchi Variabili utilizzate e Definisce le variabili e la funzione. È possibile spostare uno qualunque dei due blocchi: si selezioni ad esempio il blocco variabili utilizzate con i tasti **Shift+Freccia Giù** includendo la linea vuota al termine del blocco.

```
REM ** Variabili utilizzate **
' spremi$       stringa con spazi rimossi
' car$          un carattere della stringa data
' stringatest$ stringa data per provare FNspremi$
' k             contatore dei caratteri nel ciclo FOR-NEXT

REM ** Programma principale, Test FNspremi$ **
^cursore
  .
  .
  .
```

Si acceda al menu Modifica e si cancelli il blocco con il comando Taglia, ponendolo così nel ritaglio, si porti il cursore alla R all'inizio dell'istruzione REM del blocco Definisce le variabili e la funzione.

```
REM ** Definisce le variabili e la funzione **
^cursore
```

Si acceda nuovamente al menu Modifica, si evidenzi il comando Incolla, come illustrato nella Figura 4.3, si prema **Invio** ed il blocco Variabili utilizzate verrà inserito nella nuova posizione.

```
' Microsoft QuickBASIC 4.5  File: UQBO403.BAS
```

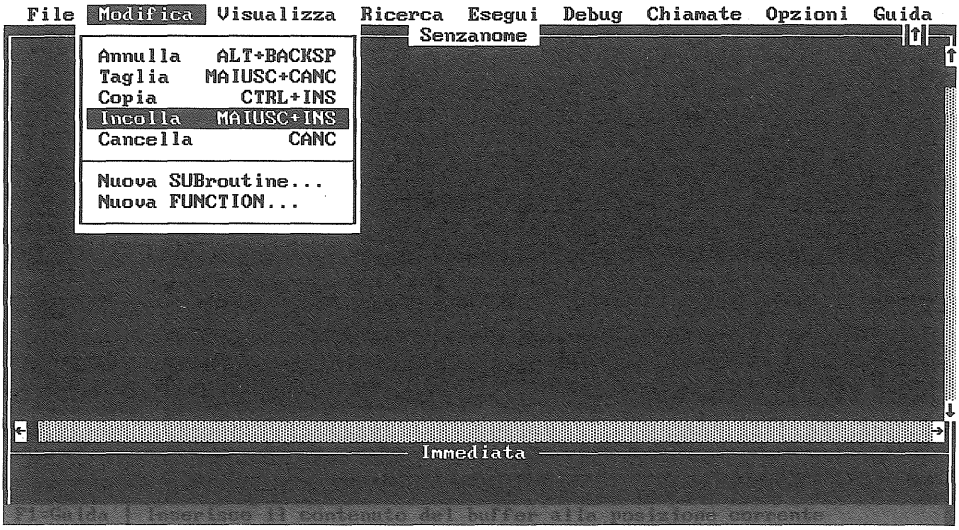


Figura 4.3 Menu Modifica con il comando Incolla evidenziato

```
REM ** Variabili utilizzate **
^cursore
' spremi$       stringa con spazi rimossi
' car$          un carattere della stringa data
' stringatest$ stringa data per provare FNspremi$
' k             contatore dei caratteri nel ciclo FOR-NEXT

REM ** Definisce le variabili e la funzione **
```

4.4.1.5 Copia

Il comando Copia esegue una copia del testo selezionato e la pone nel ritaglio; il testo nel file viene lasciato immutato.

Si selezioni come in precedenza il testo e si prema **Alt+M** per attivare il menu Modifica: si evidenzi il comando Copia e si prema **Invio**. In questo modo si pone nel ritaglio una copia del testo selezionato, mentre il testo originale rimane nella sua posizione nel file.

Si richiami nuovamente il Programma 4.3 supponendo di voler spostare il blocco Variabili utilizzate dal Programma 4.3 al Programma 4.2 e si copi il blocco nel ritaglio nel seguente modo:

- si selezioni l'intero blocco Variabili utilizzate portando il cursore alla R dell'istruzione REM al principio del blocco e premendo **Shift+Freccia Giù** sei volte in modo da evidenziare tutte le linee del blocco e quella vuota che lo segue;
- si acceda al menu File;
- si utilizzi il comando Copia per copiare il blocco nel ritaglio.

Si acceda quindi al menu File e si richiami il Programma 4.2; benché sia stato richiamato un nuovo programma, il testo che era stato copiato dal Programma 4.3 si trova ancora nel ritaglio.

Si porti il cursore alla R al principio dell'istruzione REM Programma principale, si acceda al menu Modifica, si selezioni il comando Incolla, si prema **Invio** ed il testo che si trovava nel ritaglio verrà inserito nella posizione del cursore.

```
END DEF
REM ** Variabili utilizzate **
' spremi$      stringa con spazi rimossi
' car$         un carattere della stringa data
' stringatest$ stringa data per provare FNspremi$
' k            contatore dei caratteri nel ciclo FOR-NEXT

REM ** Programma principale, Test FNspremi$ **
  .
  .
  .
```

Il blocco Variabili utilizzate è stato copiato dal Programma 4.3 al Programma 4.2.

4.4.1.6 Cancella

Il comando Cancella funziona come il comando Taglia, eccetto che il testo selezionato non viene copiato nel ritaglio, ma è cancellato e non può essere recuperato.

4.4.1.7 Altri comandi del menu Modifica

Il comando Nuova SUBroutine permette di aggiungere al programma un nuovo sottoprogramma e verrà trattato dettagliatamente nel Capitolo 5. Il comando Nuova FUNCTION

permette di aggiungere una nuova funzione al programma: anche questo comando verrà trattato dettagliatamente nel Capitolo 5.

4.4.2 Inserimento di file

Nel Capitolo 3 sono state presentate le definizioni di funzione; una di quelle memorizzate sul dischetto di strumenti è ROUNDCNT.BAS: che arrotonda al centesimo un valore in denaro.

```
REM ** Definizione della funzione Roundcnt# **
'Funzione in doppia precisione per arrotondare
'un numero in doppia precisione a due decimali
DEF FNroundcnt# (m#) = SGN(m#) * INT(100 * ABS(m#) + .5) / 100
```

Se non la si è già precedentemente memorizzata, si digiti questa breve definizione di funzione e la si memorizzi con il nome ROUNDCNT.BAS; dopo averla memorizzata è possibile includerla in qualsiasi altro programma desiderato come descritto in seguito. Si digiti innanzitutto, si digiti il seguente segmento di programma nel quale verrà inclusa la definizione di funzione:

```
REM ** VALORE DEI TITOLI CON FUNZIONE **
' Capitolo 4
' Microsoft QuickBASIC 4.5  File: UQB0404.BAS
REM ** Introduzione dati e calcolo valore **
CLS : valore# = 0: azioni = .0001
st$ = "$$###,###,###.##"
DO WHILE azioni <> 0
  INPUT "Numero di azioni "; azioni
  INPUT "Prezzo per azione"; ppa
  valore# = valore# + azioni * ppa
  PRINT
LOOP

REM ** Chiama DEF FN per stampare il valore **
PRINT "Valore totale dei titoli = ";
PRINT USING st$; FNroundcnt#(valore#)

END
```

Si porti quindi il cursore al principio della linea vuota tra il primo ed il secondo blocco della porzione di programma.

```
_' Microsoft QuickBASIC 4.5 File:  UQB0404.BAS
```

^cursore

```
REM ** Introduzione dati e calcolo  valore **
```

Questo è il punto nel quale verrà inclusa la definizione di funzione. Con il cursore in questa posizione, si acceda al menu File, si evidenzi il comandi Merge come illustrato nella Figura 4.4 e si prema **Invio**. Compare un riquadro di comunicazione per permettere l'introduzione del nome del file da includere: ci si assicuri di inserire nell'unità corretta il dischetto contenente la definizione di funzione, in modo da potervi accedere.

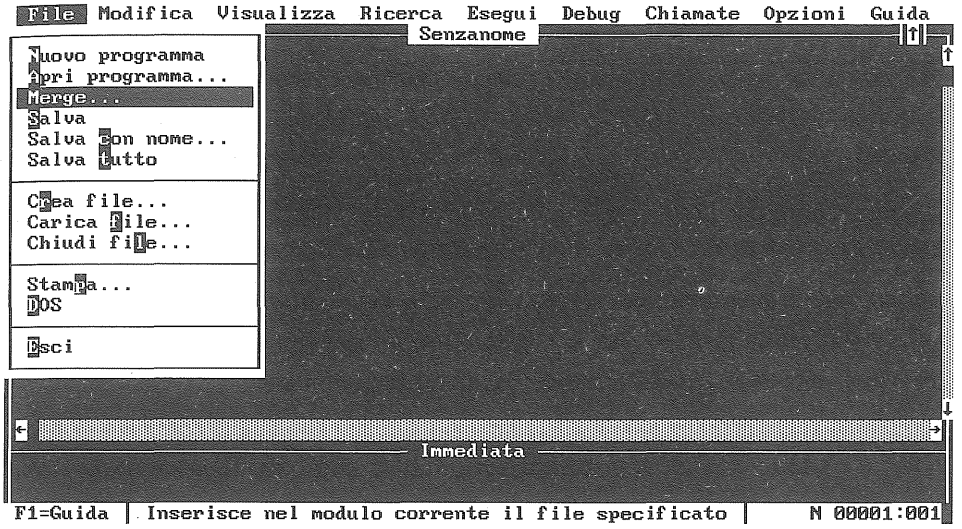


Figura 4.4 Menu File con il comando Merge evidenziato

Si digiti quindi come nome di file **ROUNDCNT.BAS**, si prema **Invio** ed il file verrà incluso nell'estratto di programma.

```
' Microsoft QuickBASIC  4.5 File: UQB0404.BAS
REM ** Definizione della funzione Roundcnt# **
'Funzione in doppia precisione per arrotondare
'un numero in doppia precisione a due decimali
DEF FNroundcnt# (m#) = SGN(m#) * INT(100 * ABS(m#) + .5) / 100

REM ** Introduzione dati e calcolo valore **
^cursore
```

Si prema **Ctrl+N** per inserire una linea di separazione tra il nuovo blocco di programma ed il blocco iniziale: in questo modo si è portata a termine l'inclusione, terminando il Programma 4.4, VALORE DEI TITOLI CON FUNZIONE.

Si memorizzi il programma con il nome UQB0404.BAS. La Figura 4.5 illustra l'esempio di esecuzione.

```
REM ** VALORE DEI TITOLI CON FUNZIONE **
' Capitolo 4
' Microsoft QuickBASIC 4.5  File: UQB0404.BAS

REM ** Definizione della funzione Roundcnt# **
'Funzione in doppia precisione per arrotondare
'un numero in doppia precisione a due decimali
DEF FNroundcnt# (m#) = SGN(m#) * INT(100 * ABS(m#) + .5) / 100

REM ** Introduzione dati e calcolo valore **
CLS : valore# = 0: azioni = .0001
st$ = "L.###.###.###"
DO WHILE azioni <> 0
  INPUT "Numero di azioni "; azioni
  INPUT "Prezzo per azione"; ppa
  valore# = valore# + azioni * ppa
  PRINT
LOOP

REM ** Chiama DEF FN per stampare il valore **
PRINT "Valore totale dei titoli = ";
PRINT USING st$; FNroundcnt#(valore#)

END
```

Programma 4.4 VALORE DEI TITOLI CON FUNZIONE

```
Numero di azioni ? 200
Prezzo per azione? 40.250

Numero di azioni ? 500
Prezzo per azione? 40.250

Numero di azioni ? 410
Prezzo per azione? 10.125

Numero di azioni ? 5
Prezzo per azione? 10.125

Numero di azioni ? 0
```

```
Prezzo per azione? 0

Valore totale dei titoli ?        L.18.236.880

Premere un tasto per continuare
```

Figura 4.5 Output del Programma 4.4

Quando si digita zero come numero di azioni ed un qualsiasi numero come prezzo per azione, il programma termina.

4.5 Sommario

In questo capitolo sono stati trattati e provati i metodi di spostamento del cursore, lo scorrimento della finestra di visualizzazione e la modifica di piccole porzioni di testo. Si è spostato il cursore tanto di un solo carattere quanto di un intero file e si è fatto scorrere il testo tanto di un sola linea quanto di un'intera finestra.

Si è appreso che una piccola porzione di testo può essere modificata come segue:

- postando il cursore, facendo scorrere lo schermo per individuare il testo da modificare, oppure con una combinazione di entrambi;
- inserendo nuovo testo o cancellando il testo esistente.

Maggiori quantità di dati sono stati modificati utilizzando i comandi del menu Modifica: si è prima selezionato il testo da modificare, si è poi scelto un comando (Taglia, Copia, Cancella od Incolla) per cancellare, copiare od inserire blocchi di testo. Si è utilizzato il ritaglio, un'area per la memorizzazione temporanea, per memorizzare temporaneamente blocchi di dati da inserire in altre posizioni del file.

Si è appreso come manipolare il testo nel file in molti modi ed anche come spostare un blocco di testo da un file ad un altro e come includere un file completo entro un altro file.

Capitolo 5

Procedure

5.1 Procedure FUNCTION
5.2 Procedure SUB
5.3 Array
5.4 Passaggio di un array ad una procedura SUB
5.5 Utilizzo di più procedure
5.6 Visualizzazione dei sottoprogrammi
5.7 Sommario

Questo capitolo introduce le procedure che, pur essendo parte di un programma QuickBASIC, sono esterne al programma principale. Una procedura può essere tanto una funzione definita da un'istruzione FUNCTION...END FUNCTION, quanto un sottoprogramma definito da un'istruzione SUB...END SUB. Un programma QuickBASIC è costituito da uno o più moduli, ciascuno dei quali equivale ad un file sorgente e possiede sempre un modulo principale che contiene il punto di entrata del programma, ovvero il punto in cui ha inizio l'esecuzione del programma. L'editor di QuickBASIC non è un semplice editor di testi, ma è anche progettato per gestire la programmazione modulare: i moduli possono contenere dei livelli sempre più bassi di codice organizzati in procedure, quali quelle definite con FUNCTION...END FUNCTION e SUB...END SUB; QuickBASIC ordina alfabeticamente questi livelli di codice e li separa dal resto del programma.

Per introdurre una procedura FUNCTION o SUB si deve utilizzare il sistema di menu completo (descritto nel Capitolo 4) il menu Modifica per crearla e Visualizza per accedervi.

5.1 Procedure FUNCTION

Una procedura FUNCTION...END FUNCTION è simile ad una funzione DEF FN...END DEF, ma presenta molti vantaggi rispetto alla più antiquata struttura di definizione di

funzione DEF FN. I principali vantaggi della programmazione con FUNCTION...END FUNCTION sono i seguenti:

- è possibile suddividere i propri programmi in unità logiche discrete ed ogni unità può essere provata e corretta con maggiore facilità rispetto ad un intero programma che non faccia uso di queste procedure;
- dopo aver portato a termine il debugging (prova e correzione) di una FUNCTION, questa può essere utilizzata come un blocco di costruzione per programmi diversi da quello per il quale era stata creata;
- i programmi che fanno uso di procedure sono generalmente più affidabili rispetto a quelli che non ne fanno uso, poiché una procedura ha uno ed un solo punto di entrata e poiché ogni variabile dichiarata al loro interno è per default locale a quella specifica procedura;
- alle procedure FUNCTION possono essere passati interi vettori.

QuickBASIC memorizza le procedure FUNCTION separatamente dal programma principale ed è possibile accedere ad esse per mezzo del menu Visualizza per la lettura e la modifica; inoltre una FUNCTION...END FUNCTION può essere posta in una Quick Library (trattata nel Capitolo 12), da dove può essere chiamata da altri programmi ogniqualvolta sia utile.

Nel Capitolo 3, si è utilizzata la DEF FN su più linee:

```
DEF FNspremi$ (stringa$)
  STATIC spremi$, k, car$
  FOR k = 1 to LEN(stringa$)
    car$ = MID(stringa$, k, 1)
    IF car$ <> " " THEN spremi$ = spremi$ + car$
  NEXT k
FNspremi$ = spremi$
END DEF
```

Questa definizione di funzione è stata utilizzata per eliminare gli spazi da una stringa e per mantenere le variabili spremi$, k e car$ locali alla funzione, era necessaria l'istruzione STATIC; la stessa definizione di funzione può essere scritta come procedura FUNCTION:

```
FUNCTION Spreme$ (stringa$)
  spremi$ = ""
  FOR k = 1 to LEN(stringa$)
    car$ = MID(stringa$, k, 1)
    IF car$ <>  " " THEN spremi$ = spremi$ + car$
  NEXT k
  Spreme$ = spremi$
END FUNCTION
```

Le variabili in una procedura FUNCTION sono sempre locali alla procedura nella quale vengono utilizzate e non è quindi necessaria l'istruzione STATIC. L'istruzione FUNCTION individua il principio della procedura FUNCTION ed include il nome della

funzione, che è un qualunque nome di variabile valido sino ad un massimo di 40 caratteri e che non può essere utilizzato contemporaneamente da un sottoprogramma (SUB) e da una FUNCTION. Nell'istruzione FUNCTION è anche incluso un elenco di variabili separate da virgole (detto elenco dei parametri), che espone il numero ed il tipo degli argomenti che devono esserle passati.

La stringa, con gli spazi rimossi, è passata nuovamente al programma principale mediante la variabile Spreme$.

END FUNCTION individua il termine della FUNCTION e ne provoca l'uscita, tornando all'istruzione immediatamente successiva a quella che aveva chiamato la FUNCTION.

È possibile disporre una o più istruzioni EXIT FUNCTION nella definizione come uscita alternativa dalla FUNCTION.

È possibile chiamare una FUNCTION allo stesso modo in cui si chiamano le funzioni predefinite di QuickBASIC (come INT e ABS) ed è possibile utilizzare il suo nome in un'espressione. La seguente funzione predefinita stampa in maiuscolo ogni carattere di stringatest$:

```
PRINT UCASE$(stringatest$)
```

La seguente funzione definita dall'utente stampa il valore di stringatest$ dopo che la FUNCTION Spreme$ ne ha rimosso tutti gli spazi:

```
PRINT Spreme$(stringatest$)
```

Il programma in questo capitolo è simile al programma SPREMISTRINGHE del Capitolo 3, tuttavia questo programma si avvale della procedura FUNCTION anziché della DEF FN su più linee ed inoltre utilizza una uscita alternativa dal DO...LOOP quando si preme il tasto Esc. La sezione principale del programma è:

```
REM ** SPREMISTRINGHE CON FUNCTION **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0501.BAS

REM ** Programma principale **
' Prende la stringa di testo, chiama la FUNCTION
' Stampa la stringa compressa
' Premere Esc per terminare, un altro tasto per continuare

CLS : DEFINT A-Z
DO
  LINE INPUT "Stringa, prego: "; stringatest$
  PRINT Spreme$(stringatest$)          'Chiama FUNCTION & stampa
  PRINT
  PRINT "Premere Esc per finire od un altro tasto per continuare"
  PRINT : ta$ = INPUT$(1)
  IF ta$ = CHR$ (27) THEN EXIT DO
LOOP

END
```

Si digiti questa sezione e la si memorizzi con il nome UQB0501.BAS. La FUNCTION che deve essere aggiunta ha nome Spreme$ ed è chiamata, con il relativo argomento (stringatest$), in un'istruzione PRINT del DO LOOP del modulo principale del programma.

```
PRINT Spreme$(stringatest$)
```

La funzione non compare come parte del modulo principale illustrato, ma viene digitata dopo aver introdotto e memorizzato la sezione principale del programma.

Dopo aver memorizzato la sezione del programma, si acceda al menu Modifica e si evidenzi il comando Nuova FUNCTION, come illustrato nella Figura 5.1, si prema il tasto **Invio** e verrà visualizzato un riquadro di comunicazione, illustrato nella Figura 5.2.

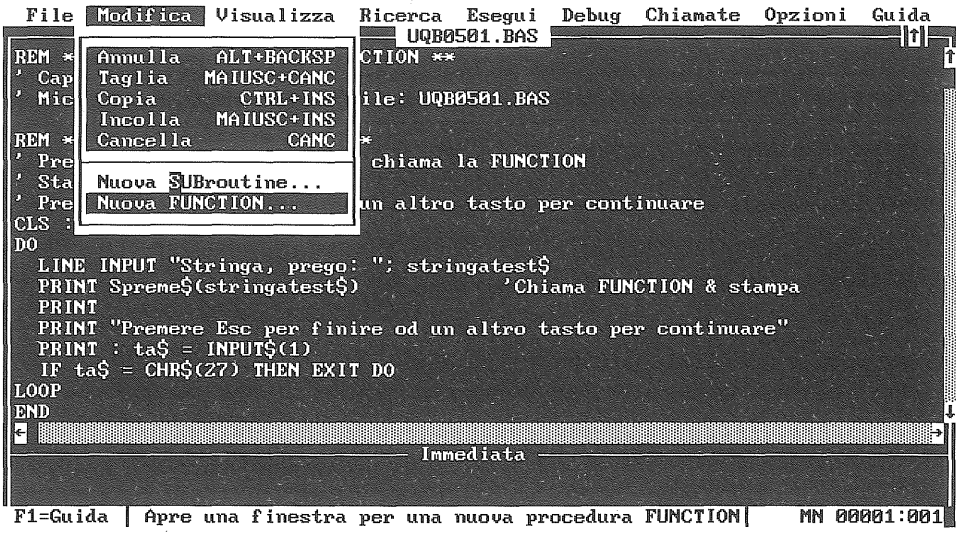


Figura 5.1 Menu Modifica con Nuova FUNCTION evidenziato

Si digiti il nome della nuova FUNCTION: **Spreme$** e verrà visualizzata una nuova finestra contenente già le istruzioni di principio e termine della FUNCTION.

```
FUNCTION Spreme$_
END FUNCTION
```

Si osservi che il cursore si trova sulla destra del nome della FUNCTION in modo da poter agevolmente aggiungere a questo l'argomento che deve essere passato (stringa$).

```
FUNCTION Spreme$ (stringa$)_
```

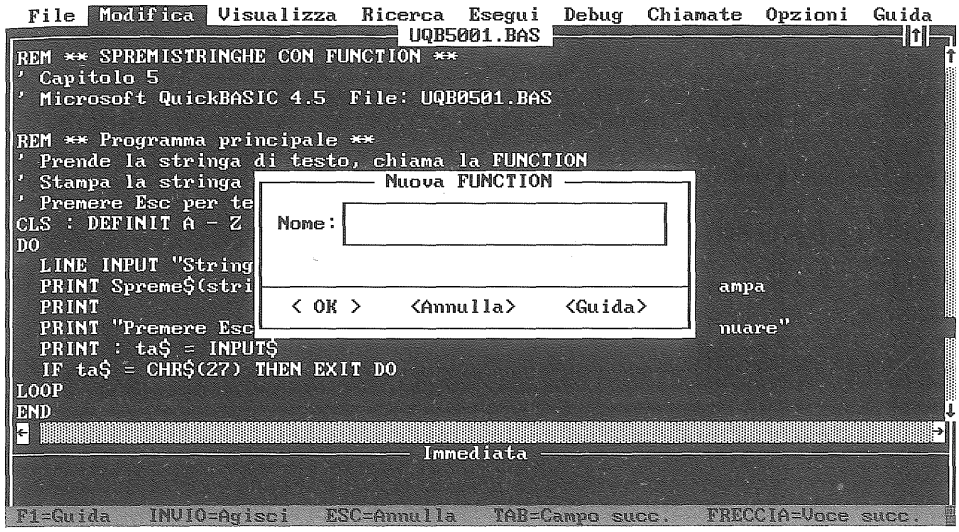


Figura 5.2 Riquadro di comunicazione di Nuova FUNCTION

Si prema **Invio** e si digiti il resto della definizione della funzione, come illustrato sotto. Ogniqualvolta si preme **Invio** viene creata una nuova linea vuota, pronta per l'istruzione successiva.

```
FUNCTION Spreme$ (stringa$)
' Elimina gli spazi da una stringa
' che viene introdotta nel programma principale
  spremi$ = ""
  FOR k = 1 TO LEN(stringa$)
    car$ = MID$(stringa$, k, 1)
    IF car$ <> " " THEN spremi$ = spremi$ + car$
  NEXT k
  Spreme$ = spremi$            'assegna un valore di ritorno
END FUNCTION
```

Quando si è terminata l'introduzione della FUNCTION, si acceda al menu File e si evidenzi il comando Salva; nell'esempio verranno memorizzati insieme il programma principale e la FUNCTION con il nome UQB0501.BAS. Questo completa il Programma 5.1, SPREMI-STRINGHE CON FUNCTION.

Si acceda nuovamente al programma principale dal menu Visualizza e si osservi che QuickBASIC ha aggiunto la seguente linea nel principio del programma:

```
DECLARE FUNCTION Spreme$ (stringa$)
```

L'istruzione DECLARE è indispensabile al programma per poter chiamare procedure che sono definite più avanti nel modulo, come FUNCTION...END FUNCTION o SUB...END

SUB o procedure definite insieme in un altro modulo; l'istruzione DECLARE rimane parte del programma principale.

```
DECLARE FUNCTION Spreme$ (stringa$)
REM ** SPREMISTRINGHE CON FUNCTION **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0501.BAS

REM ** Programma principale **
' Prende la stringa di testo, chiama la FUNCTION
' Stampa la stringa compressa
' Premere Esc per terminare, un altro tasto per continuare

CLS : DEFINT A-Z
DO
  LINE INPUT "Stringa, prego: "; stringatest$
  PRINT Spreme$(stringatest$)          'Chiama FUNCTION & stampa
  PRINT
  PRINT "Premere Esc per finire od un altro tasto per continuare"
  PRINT : ta$ = INPUT$(1)
  IF ta$ = CHR$ (27) THEN EXIT DO
LOOP

END

FUNCTION Spreme$ (stringa$)
' Elimina gli spazi da una stringa
' che viene introdotta nel programma principale

  spremi$ = ""
  FOR k = 1 TO LEN(stringa$)
    car$ = MID$(stringa$, k, 1)
    IF car$ <>  " " THEN spremi$ = spremi$ + car$
  NEXT k
  Spreme$ = spremi$            'assegna un valore di ritorno
END FUNCTION

DECLARE SUB Spreme (stringa$, spremi$)
REM ** SPREMISTRINGHE CON SOTTOPROGRAMMA **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0502.BAS

REM ** Programma principale **
' Prende la stringa, chiama la SUB
' Stampa la stringa compressa
' Premere Esc per terminare, un altro tasto per continuare

CLS : DEFINT A-Z
DO
```

```
  LINE INPUT "Stringa, prego: "; stringatest$
  CALL Spreme(stringatest$, spremi$)
  PRINT spremi$: PRINT
  PRINT "Premere Esc per finire od un altro tasto per continuare"
  PRINT : ta$ = INPUT(1)
  IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END

SUB Spreme (stringa$, spremi$)
' Elimina gli spazi da una stringa
' che viene introdotta nel programma principale
  spremi$ = ""
  FOR k = 1 TO LEN(stringa$)
    car$ = MID$(stringa$, k, 1)
    IF car$ <> " " THEN spremi$ = spremi$ + car$
  NEXT k
END SUB
```

Programma 5.1 SPREMISTRINGHE CON FUNCTION

Si esegua il Programma 5.1 per assicurarsi che funzioni come il programma che faceva uso di DEF FN nel Capitolo 3. La Figura 5.3 illustra una tipica esecuzione del Programma 5.1

```
Stringa, prego: Togli tutti gli spazi da questa stringa.
Toglituttiglispazidaquestastringa.


Premere Esc per finire od un altro tasto per continuare


Stringa, prego: 02 7000 18 45
0270001845


Premere Esc per finire od un altro tasto per continuare


Stringa, prego: Via Fiamma 22, 20100 Milano
ViaFiamma22,20100Milano


Stringa, prego:_
                ^cursore
```

Figura 5.3 Output del Programma 5.1

5.2 Procedure SUB

Se si sono utilizzate versioni del BASIC delle prime generazioni, si noteranno delle somiglianze tra un sottoprogramma di QuickBASIC (SUB...END SUB) e la subroutine, utilizzata dalle prime versioni di BASIC e che può essere utilizzata anche da QuickBASIC. La subroutine (chiamata da un'istruzione GOSUB) contiene solamente variabili globali, mentre la struttura di sottoprogramma di QuickBASIC permette l'uso tanto di variabili globali quanto di variabili locali ed offre inoltre anche altri vantaggi che saranno individuati attraverso il suo uso.

Si troveranno delle somiglianze anche tra le procedure SUB e FUNCTION: ad esempio, SUB...END SUB ha una struttura simile a quella di FUNCTION...END FUNCTION. La FUNCTION Spreme$, vista nel Programma 5.1, può essere scritta come procedura SUB nel seguente modo:

```
SUB Spreme (stringa$, spremi$)
' Elimina gli spazi da una stringa
' che viene introdotta nel programma principale
  spremi$ = ""
  FOR k = 1 TO LEN(stringa$)
    car$ = MID$(stringa$, k, 1)
    IF car$ <> " " THEN spremi$ = spremi$ + car$
  NEXT k
END SUB
```

L'istruzione SUB individua il principio della procedura SUB: include il nome del sottoprogramma (Spreme$) ed un elenco di variabili scambiate tra programma principale e sottoprogramma. In questo caso l'elenco include le variabili stringa$ (passata dal programma principale al sottoprogramma) e la variabile spremi$ (restituita dal sottoprogramma al programma principale); è anche possibile scambiare più variabili.

Non è ammesso usare lo stesso nome per una procedura SUB e per una FUNCTION ed inoltre i nomi di SUB e FUNCTION devono differire da quelli di qualunque altra variabile utilizzata.

END SUB individua il termine della procedura SUB e ne comporta l'uscita: l'esecuzione torna all'istruzione immediatamente precedente a quella che aveva chiamato la SUB; è possibile disporre nel corpo della definizione una o più istruzioni EXIT SUB come metodo alternativo per l'uscita.

Dopo essere stato definito un sottoprogramma viene chiamato in modo differente rispetto ad una procedura FUNCTION. Esistono due metodi per chiamare una SUB:

- porre il suo nome in una istruzione CALL, ad esempio:

  ```
  CALL Spreme(stringatest$, spremi$)
  ```

- utilizzare il suo nome come istruzione, come nel caso:

```
Spreme stringatest$$, spremi$
```

Spreme è il nome di un sottoprogramma, stringatest$ e spremi$ sono gli argomenti passati da programma a sottoprogramma e viceversa.

Se si fa uso della parola chiave CALL, è necessario porre l'argomento che deve essere passato tra parentesi, mentre se si omette CALL le parentesi non sono necessarie. Nel seguito del libro tutte le chiamate a sottoprogrammi faranno uso dell'istruzione CALL per esprimere chiaramente che ci si sta riferendo ad un sottoprogramma.

Il programma che segue contiene un modulo principale che illustra l'uso di sottoprogrammi in sostituzione a FUNCTION...END FUNCTION.

```
REM ** SPREMISTRINGHE CON SOTTOPROGRAMMA **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0502.BAS

REM ** Programma principale **
' Prende la stringa, chiama la SUB
' Stampa la stringa compressa
' Premere Esc per terminare, un altro tasto per continuare
CLS : DEFINT A-Z
DO
  LINE INPUT "Stringa, prego: "; stringatest$
  CALL Spreme(stringatest$, spremi$)
  PRINT spremi$: PRINT
  PRINT "Premere Esc per finire od un altro tasto per continuare"
  PRINT : ta$ = INPUT(1)
  IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END
```

Si osservi la differenza nel modo in cui vengono chiamati il sottoprogramma e la FUNCTION: innanzitutto viene la parola chiave CALL, poi il nome del sottoprogramma, quindi l'argomento che deve essere passato al sottoprogramma (stringatest$) ed infine quello che deve essere restituito dal sottoprogramma (spremi$); l'argomento restituito è poi stampato da un'istruzione separata.

Si osservi che nel nome del sottoprogramma non si pongono simboli di dichiarazione di tipo di dato (come $).

Si digiti il modulo principale del nuovo programma e lo si memorizzi in formato testo con il nome UQB0502.BAS, si acceda al menu Modifica, come si era fatto per il Programma 5.1, ma questa volta si evidenzi il comando Nuova SUBroutine e si prema quindi **Invio**: comparirà un riquadro di comunicazione che richiede il nome del sottoprogramma e quindi si digiti il nome **Spreme**.

Nome: Spreme

Si prema **Invio** e compariranno il principio ed il termine del sottoprogramma; si completi come segue la prima linea:

```
SUB Spreme (stringa$, spremi$)
```

Ora si completi il sottoprogramma come segue:

```
SUB Spreme (stringa$, spremi$)
' Elimina gli spazi da una stringa
' che viene introdotta nel programma principale
  spremi$ = ""
  FOR k = 1 TO LEN(stringa$)
    car$ = MID$(stringa$, k, 1)
    IF car$ <> " " THEN spremi$ = spremi$ + car$
  NEXT k
END SUB
```

Quando si è terminata l'introduzione del sottoprogramma, si acceda al menu File, si evidenzi Salva e si prema Invio: in questo modo si completa il Programma 5.2, SPREMI-STRINGHE CON SOTTOPROGRAMMA.

Si acceda al programma principale di UQB0502.BAS e si osservi che è stata aggiunta un'istruzione DECLARE come linea iniziale. Se si sta utilizzando l'editor di QuickBASIC per scrivere i programmi, l'istruzione DECLARE per le procedure verrà automaticamente inserita e questo è un altro valido motivo per chiamare i sottoprogrammi con la parola chiave CALL. Si esegua alcune volte il Programma 5.2 per assicurarsi che funzioni come il Programma 5.1; l'output dovrebbe presentarsi simile a quello nella Figura 5.3.

```
DECLARE SUB Spreme (stringa$, spremi$)
REM ** SPREMISTRINGHE CON SOTTOPROGRAMMA **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0502.BAS

REM ** Programma principale **
' Prende la stringa, chiama la SUB
' Stampa la stringa compressa
' Premere Esc per terminare, un altro tasto per continuare
CLS : DEFINT A-Z
DO
  LINE INPUT "Stringa, prego: "; stringatest$
  CALL Spreme(stringatest$, spremi$)
  PRINT spremi$: PRINT
  PRINT "Premere Esc per finire od un altro tasto per continuare"
  PRINT : ta$ = INPUT$(1)
  IF ta$ = CHR$(27) THEN EXIT DO
LOOP

END

SUB Spreme (stringa$, spremi$)
' Elimina gli spazi da una stringa
```

```
' che viene introdotta nel programma principale
  spremi$ = ""
  FOR k = 1 TO LEN(stringa$)
    car$ = MID$(stringa$, k, 1)
    IF car$ <> " " THEN spremi$ = spremi$ + car$
  NEXT k
END SUB
```

Programma 5.2 SPREMISTRINGHE CON SOTTOPROGRAMMA

5.3 Array

Il BASIC originale di Dartmouth disponeva solamente di array mono e bidimensionali: nella documentazione gli array monodimensionali erano chiamati liste o vettori e quelli bidimensionali tavole o matrici. Veniva utilizzata l'istruzione DIM per specificare il limite massimo per gli indici ed il limite inferiore era sempre 1; se non compariva l'istruzione DIM, il BASIC poneva per default il limite superiore pari a 10. QuickBASIC permette array numerici o di stringhe fino a 63 dimensioni. Gli array numerici possono essere di qualunque tipo: intero, intero lungo, singola precisione o doppia precisione.

5.3.1 Dimensionamento degli array

In QuickBASIC è possibile impostare i limiti inferiore e superiore per gli indici degli array con l'istruzione DIM; è anche possibile impostare il limite inferiore con l'istruzione OPTION BASE.

```
OPTION BASE 1    'pone il limite inferiore degli indici uguale a 1
OPTION BASE 0    'pone il limite inferiore degli indici uguale a 0
```

Il valore di default per il limite inferiore è 0: si raccomanda di non utilizzare OPTION BASE per variarlo ad 1 se non se ne ha assolutamente una buona ragione. L'istruzione DIM inizializza a zero tutti gli elementi degli array numerici ed a stringa nulla, o vuota, "" gli elementi degli array di stringhe; dovrebbe essere utilizzata prima del'array a cui si riferisce. È buona norma porre tutte le istruzioni DIM nella parte iniziale del programma, al di fuori di ogni ciclo in modo da non incorrere tanto facilmente in un errore di "array già dimensionato".

È possibile dimensionare un array come STATIC (allocato durante la compilazione del programma) oppure come DYNAMIC (allocato quando il programma viene eseguito): gli array dimensionati con sole costanti numeriche sono STATIC, mentre quelli dimensionati con variabili sono dinamici. Gli array STATIC hanno dimensioni stabilite nel momento in

cui il programma viene tradotto da QuickBASIC; la Tabella 5.1 illustra alcuni esempi di istruzioni DIM per il dimensionamento di array STATIC: la parola chiave TO, in Stato$(1 TO 60) e Tris%(1 TO 3, 1 TO 3), definisce il limite sia inferiore che superiore degli indici.

Tabella 5.1 Istruzioni DIM utilizzate per il dimensionamento di array STATIC

Istruzione DIM	Descrizione dell'array STATIC
DIM prezzo(100)	Array monodimensionale in singola precisione con 101 elementi: da prezzo(0) a prezzo(100).
DIM Stato$(1 TO 60)	Array monodimensionale intero con 60 elementi: da Stato$(1) a Stato$(60). Si noti l'uso di TO per impostare sia il limite inferiore che quello superiore.
DIM Tris%(1 TO 3, 1 TO 3)	Array bidimensionale intero con 9 elementi; lo si può pensare come composto da tre linee e tre colonne.
DIM Iperspazio#(99,99,99,99)	Array quadridimensionale in doppia precisione con 100*100*100*100 elementi.

Gli array DYNAMIC utilizzano nell'istruzione DIM delle variabili come indici; la Tabella 5.2 illustra degli esempi di dimensionamento di array DYNAMIC.

QuickBASIC possiede due funzioni che facilitano la determinazione dei limiti inferiore e superiore e permettono la scrittura di sottoprogrammi e FUNCTION generali utilizzabili da qualunque altro programma che faccia uso di array.

LBOUND restituisce il limite inferiore (il più piccolo indice disponibile di un array) e viene utilizzata nella forma:

```
LBOUND (Celsius, 1)
```

dove Celsius è il nome dell'array ed 1 il numero di dimensioni dell'array (massimo 63). Il numero utilizzato indica quale dimensione si desidera che venga restituita. UBOUND restituisce il limite superiore (il più grande indice disponibile di un array) ed è utilizzata nella forma:

```
UBOUND (Celsius, 1)
```

dove Celsius è il nome dell'array ed 1 è il numero di dimensioni nell'array (massimo 63). Il numero utilizzato indica quale dimensione si desidera che venga restituita. Si consideri ad esempio un array dimensionato nel seguente modo:

```
DIM Cassettiera$ (1 TO 100, 0 TO 25, -5 TO 5)
```

Tabella 5.2 Istruzioni DIM utilizzate per il dimensionamento di array DYNAMIC

Istruzione DIM	Descrizione dell'array DYNAMIC
DIM prezzo(Ultimo%)	Array monodimensionale in singola precisione con elementi da prezzo(0) a prezzo(Ultimo%).
DIM prezzo(1 TO Ultimo%)	Array monodimensionale in singola precisione con elementi da prezzo(1) a prezzo(Ultimo%).
DIM prezzo(10 TO Ultimo%)	Array monodimensionale in singola precisione con elementi da prezzo(10) a prezzo(Ultimo%).
DIM Gomoku%(1 TO n%, 1 TO n%)	Array bidimensionale intero con elementi da Gomoku%(1,1) a Gomoku%(n%,n%).
DIM postaaerea%(n%,n%)	Array bidimensionale intero con elementi da postaaerea%(0,0) a postaaerea%(n%,n%).

La Tabella 5.3 illustra i valori che dovrebbero essere restituiti dalle istruzioni LBOUND ed UBOUND specificate.

Tabella 5.3 Valori restituiti dalle istruzioni LBOUND ed UBOUND

Istruzione	Valore restituito
LBOUND(Cassettiera$, 1)	1
LBOUND(Cassettiera$, 2)	0
LBOUND(Cassettiera$, 3)	-5
UBOUND(Cassettiera$, 1)	100
UBOUND(Cassettiera$, 2)	25
UBOUND(Cassettiera$, 3)	5

È possibile servirsi della forma abbreviata di LBOUND ed UBOUND per array monodimensionali, come Celsius nell'esempio precedente, poiché il valore di default per la dimensione è 1 (array monodimensionale).

```
LBOUND (Celsius) anziche' LBOUND (Celsius, 1)
UBOUND (Celsius) anziche' UBOUND (Celsius, 1)
```

Se l'array non viene dimensionato, LBOUND, nel caso che non venga impostato ad 1 con OPTION BASE il limite inferiore, è pari a 0; è bene servirsi di una istruzione DIM ogniqualvolta si utilizzi un array in modo da non fare confusione con gli indici.

QuickBASIC rende semplice l'impostazione dei limiti superiore ed inferiore in modo chiaro, per cui si faccia uso di tutti i metodi disponibili per per rendere il programma di facile lettura e comprensione. In seguito ci si servirà in alcuni esempi degli strumenti sin qui trattati e se ne introdurranno di nuovi.

5.3.2 Passaggio di array a procedure FUNCTION

È possibile passare interi array a procedure FUNCTION o SUB ponendo nell'elenco degli argomenti il nome dell'array seguito da una coppia di parentesi vuote (sinistra e destra); anche nell'elenco dei parametri delle istruzioni FUNCTION o SUB deve comparire il nome dell'array seguito da parentesi vuote. Il Programma 5.3, SOMME E MEDIA DEGLI ELEMENTI DI ARRAY STATICI, si serve di una procedura FUNCTION per calcolare la somma degli elementi in un array.

```
DECLARE FUNCTION SommaArray% (array%())
REM ** SOMMA E MEDIA DEGLI ELEMENTI DI ARRAY STATICI **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0503.BAS

' Dimensiona gli array come STATIC, definisce gli interi
DEFINT A-Z
DIM Celsius(1 TO 7), Fahrenheit(1 TO 12)
' legge i dati dall'array Celsius
FOR k = 1 TO 7
  READ Celsius(k)
NEXT k
DATA 20, 18, 23, 22, 19, 21, 23

'Legge i dati per l'array Fahrenheit
FOR k = 1 TO 12
  READ Fahrenheit(k)
NEXT k
DATA 68, 67, 73, 71, 79, 82, 89, 85, 80, 75, 76, 74

' Stampa la somma e media per l'array Celsius
CLS
PRINT "Somma Celsius: "; SommaArray%(Celsius())
PRINT "Media Celsius: "; SommaArray%(Celsius()) / 7

' Stampa somma e media del'array Fahrenheit
PRINT "Somma Fahrenheit: "; SommaArray%(Fahrenheit())
PRINT "Media Fahrenheit: "; SommaArray%(Fahrenheit()) / 12
```

```
END

FUNCTION SommaArray% (array())
'Restituisce la somma degli elementi
'di un array monodimensionale
   somma = 0
   FOR k = LBOUND(array) TO UBOUND(array)
     somma = somma + array(k)
   NEXT k
   SommaArray% = somma
END FUNCTION
```

Programma 5.3 SOMMA E MEDIA DEGLI ELEMENTI DI ARRAY STATICI

Nel modulo principale del Programma 5.3 vengono creati due array: Celsius() è un array monodimensionale con 7 elementi e Farhenheit() è un array monodimensionale con 12 elementi; gli elementi nei due array vengono passati alla FUNCTION SommaArray dalle istruzioni:

```
PRINT "Somma Celsius: "; SommaArray%(Celsius())
PRINT "Somma Fahrenheit: "; SommaArray$(Farhenheit())
```

Si osservi che l'istruzione PRINT contiene i nomi della FUNCTION (SommaArray%) e dell'array (Celsius o Farhenheit) seguiti da una coppia di parentesi vuote, che sono essenziali per passare gli elementi dell'array.

Nell'istruzione FUNCTION compare un nome di array anch'esso seguito da una coppia di parentesi vuote, che sono necessarie per passare gli elementi dell'array.

```
FUNCTION SommaArray% (array())
```

La somma degli elementi dell'array viene restituita al programma principale dalla FUNCTION con il seguente assegnamento:

```
SommaArray% = somma
```

In aggiunta alla stampa della somma degli elementi di ciascun array il modulo principale stampa la media degli elementi di ogni array.

```
PRINT "Media Celsius: "; SommaArray%(Celsius()) /7
PRINT "Media Farhenheit: "; SommaArray$(Farhenheit()) / 12
```

La FUNCTION viene chiamata due volte per ogni array: la prima volta viene stampata la somma degli elementi e la seconda volta la somma degli elementi viene utilizzata per determinare la loro media. Se l'array avesse grandi dimensioni si potrebbe voler chiamare la FUNCTION una sola volta: sarebbe possibile ottenere la somma dalla funzione, assegnare il valore ad una variabile e calcolare la media dei suoi elementi. Si osservi che con LBOUND ed UBOUND, la FUNCTION può gestire array di dimensioni diverse. La Figura 5.4 illustra come il Programma 5.3 stampa le somme delle temperature e le loro medie: le medie sono stampate in singola precisione, l'espressione che calcola le medie nell'istruzione PRINT contiene una divisione per i valori numerici 7 e 12 e QuickBASIC tratta il risultato come un valore in singola precisione, stampandolo in questo formato.

```
Somma Celsius:  146
Media Celsius:  20.85714
Somma Fahrenheit:  919
Media Fahrenheit:  76.58334

Premere un tasto per continuare
```

Figura 5.4 Output del Programma 5.3

Si osservi l'istruzione DECLARE che QuickBASIC ha aggiunto nella parte iniziale del Programma 5.3: si noti che compare anche in questa il tipo di dati utilizzato nel creare la FUNCTION.

```
DECLARE FUNCTION SommaArray% (array%())
```

Il tipo di dato (%) specificato nell'istruzione DECLARE compare tanto al seguito del nome della FUNCTION (SommaArray%), quanto al seguito del nome dell'array (array%).

Il Programma 5.4, SOMMA E MEDIA DEGLI ELEMENTI DI UN ARRAY DINAMICO, mostra l'uso degli array dinamici e dell'istruzione REDIM, che permette di ridimensionare un array in qualunque momento durante l'esecuzione di un programma.

```
DECLARE FUNCTION SommaArray! (array!())
REM ** SOMMA E MEDIA DEGLI ELEMENTI DI UN ARRAY DINAMICO **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0504.BAS
' Array monodimensionali introdotti da tastiera

' Si introduca il numero di elementi nell'array
CLS : DEFINT A-Z
INPUT "Quanti elementi"; dimensione

' Ridimensiona l'array dinamico.
REDIM array!(1 TO dimensione)

' Introdurre gli elementi da array!(1) ad array!(dimensione)
FOR k = 1 TO UBOUND(array!)
   PRINT "Elemento numero"; k; : INPUT array!(k)
NEXT k

' Stampa la somma e la media degli elementi
PRINT
PRINT "Somma degli elementi: "; SommaArray!(array!())
PRINT "Media degli elementi: "; SommaArray!(array!()) / dimensione

END

 FUNCTION SommaArray! (array!())
```

```
'Restituisce la somma degli elementi
'di un array monodimensionale
  somma! = 0
  FOR k = LBOUND(array!) TO UBOUND(array!)
    somma! = somma! + array!(k)
  NEXT k
  SommaArray! = somma!
END FUNCTION
```

Programma 5.4 SOMMA E MEDIA DEGLI ELEMENTI DI UN ARRAY DINAMICO

Quando un programma viene compilato da QuickBASIC tutti gli array dichiarati in un'istruzione REDIM vengono trattati come array dinamici, mentre quando si avvia il programma e l'istruzione REDIM viene eseguita, l'array viene deallocato (se era stato precedentemente allocato) e l'istruzione REDIM rialloca l'array con le nuove dimensioni specificate. I precedenti valori degli elementi dell'array sono perduti poiché tutti gli elementi numerici vengono inizializzati a zero e gli elementi stringa a stringa nulla (vuota). La Figura 5.5 illustra una tipica esecuzione del Programma 5.4.

```
Quanti elementi? 4

Elemento numero 1 ? 14.2

Elemento numero 2 ? 17.65

Elemento numero 3 ? 22

Elemento numero 4 ? 33

Somma degli elementi: 86.25

Media degli elementi: 21.7125


Premere un tasto per continuare
```

Figura 5.5 Output del Programma 5.4

Il Programma 5.5, VALORE LORDO DEI TITOLI, mostra l'uso della FUNCTION SubTotale#, che calcola la somma dei prodotti degli elementi corrispondenti di due array. Sono possibili parecchi altri usi per questa funzione.

```
DECLARE FUNCTION SubTotale# (Quantita&(), PrezzoUnitario#())
REM ** VALORE LORDO DEI TITOLI **
 Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0505.BAS
' Inizializza ed introduce il numero di elementi nell'array
CLS : DEFINT A-Z
INPUT "Quanti elementi (coppie di numeri)"; dimensione

' Ridimensiona gli array dinamici, Azioni& e PrezzoAzione#
' Azioni come intero lungo, PrezzoAzione come doppia precisione
```

```
REDIM Azioni&(1 TO dimensione), PrezzoAzione#(1 TO dimensione)

'Introdurre dati del titolo: numero di azioni
'e prezzo per azione
FOR k = 1 TO UBOUND(Azioni&)
   PRINT
   INPUT "Numero di azioni"; Azioni&(k)
   INPUT "Prezzo per azione"; PrezzoAzione#(k)
NEXT k
PRINT
PRINT "Il valore lordo e' "; SubTotale#(Azioni&(), PrezzoAzione#())

END

FUNCTION SubTotale# (Quantita&(), PrezzoUnitario#())
' Restituisce la somma dei prodotti di elementi corrispondenti
' Assume che entrambe gli array abbiano gli stessi LBOUND, UBOUND
   Primo = LBOUND(Quantita&)
   Ultimo = UBOUND(Quantita&)
   somma# = 0
   FOR k = Primo TO Ultimo
     somma# = somma# + Quantita&(k) * PrezzoUnitario#(k)
   NEXT k
   SubTotale# = somma#
END FUNCTION
```

Programma 5.5 VALORE LORDO DEI TITOLI

5.4 Passaggio di un array ad una procedura SUB

È possibile passare un array ad una procedura SUB ponendo nell'elenco degli argomenti di una istruzione CALL il nome dell'array seguito da una coppia di parentesi vuote; un array corrispondente deve comparire anche nell'elenco dei parametri dell'istruzione SUB che indica il principio della procedura. Una procedura SUB può restituire un intero array al modulo del programma che aveva chiamato la procedura.

Il Programma 5.6, CREA UN ARRAY NUMERICO DISORDINATO, crea un array di interi da 1 ad n ed in seguito rimescola l'array utilizzando un sottoprogramma con nome Mescola, che, poiché crea un array rimescolato in cui ogni elemento è noto (interi da 1 ad n), è utile per provare, come si vedrà, procedure di ricerca e riordinamento.

```
DECLARE SUB Mescola (array%())
REM ** CREA UN ARRAY NUMERICO DISORDINATO **
```

```
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0506.BAS

'Genera un array di interi da 1 ad n, poi lo rimescola.
CLS : DEFINT A-Z

DO
   PRINT "L'array iniziale e' una lista ordinata da 1 ad n."
   INPUT "Indicare il valore di n desiderato"; n

   ' Ridimensiona Numeri come array DYNAMIC
   ' monodimensionale di interi
   REDIM Numeri%(1 TO n)

   ' Crea un array ordinato con l'elemento k-esimo uguale a k.
   FOR k = 1 TO n
     Numeri%(k) = k
   NEXT k

   ' Mescola l'array
   CALL Mescola(Numeri%())

   ' Stampa l'array mescolato.
   PRINT
   PRINT "Contenuti dell'array mescolato:"
   PRINT
   FOR k = 1 TO n
     PRINT Numeri%(k)
   NEXT k
   PRINT
   PRINT "Premere Esc per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP
END

SUB Mescola (array%())
' Mescola casualmente un array numerico
' Assume che il limite inferiore sia 1.
   RANDOMIZE (TIMER)
   ' Trova la dimensione dell'array (numero di elementi)
   dimensione = UBOUND(array%)
   ' Scambia ogni elemento con uno selezionato casualmente
   FOR k = 1 TO dimensione
     indicecas = INT(dimensione * RND(1)) + 1
     SWAP array%(k), array%(indicecas)
   NEXT k
END SUB
```

Programma 5.6 CREA UN ARRAY NUMERICO DISORDINATO

La Figura 5.6 illustra una esecuzione di prova per due valori di n: si noti che ad ogni esecuzione tutti i numeri interi da 1 a n compaiono nella lista mescolata prodotta dal programma.

```
L'array iniziale e' una lista ordinata da 1 ad n.
Indicare il valore di n desiderato? 15

Contenuti dell'array mescolato:
11          2           13          7           9
5           15          10          4           1
12          8           6           14          3

Premere Esc per finire od un altro tasto per continuare
L'array iniziale e' una lista ordinata da 1 ad n.
Indicare il valore di n desiderato? 25

Contenuti dell'array mescolato:
9           3           11          18          1
17          7           14          5           23
22          16          4           24          13
15          20          25          21          2
19          8           12          6           10

Premere Esc per finire od un altro numero per continuare
```

Figura 5.6 Output di due esecuzioni del Programma 5.6

Il seguente sottoprogramma ordina un array di interi: si osservino l'uso di DO...LOOP con una istruzione WHILE ed il blocco con la struttura IF...THEN.

```
SUB BubbleSort (array%())
' Riordina un array di interi da cima a fondo

  cima = LBOUND(arry%)
  fondo = UBOUND(array%)

  DO WHILE cima  fondo
    FOR qui = fondo TO cima + 1 STEP -1
     IF array%(qui)  array%(qui - 1) THEN
       SWAP array%(qui), array%(qui - 1)
```

```
    END IF
   NEXT qui
   cima = cima + 1
  LOOP
END SUB
```

Il Programma 5.7, PROGRAMMA DI PROVA DI BUBBLESORT crea un array di interi da 1 ad n, mescola l'array, stampa l'array mescolato, lo riordina e stampa il tempo necessario per il riordinamento dell'array.

```
DECLARE SUB Mescola (array%())
DECLARE SUB StampaArray (array%())
DECLARE SUB BubbleSort (array%())
REM ** PROGRAMMA DI PROVA DI BUBBLESORT **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0507.BAS

'Inizializza
CLS : DEFINT A-Z

DO
   ' determina dimensione dell'array e ridimensiona Numeri
   ' come array STATIC di interi
   INPUT "Dimensione dell'array"; n
   REDIM Numeri%(1 TO n)

   ' Crea un array disordinato contenente gli interi da 1 a n.
   FOR k = 1 TO n
     Numeri%(k) = k
   NEXT k
   CALL Mescola(Numeri%())

   ' Stampa l'array mescolato
   PRINT : PRINT "Array mescolato:"
   CALL StampaArray(Numeri%())

   ' Riordina l'array mescolato. Prende il tempo.
   inizio# = TIMER
   CALL BubbleSort(Numeri%())
   termine# = TIMER

   ' Stampa l'array riordinato ed il tempo di riordinamento
   PRINT "Array ordinato:"
   CALL StampaArray(Numeri%())
   PRINT "Tempo di ordinamento:"; termine# - inizio#; "secondi"

   ' Sceglie se terminare o continuare
   PRINT : PRINT "Premere Esc per finire od un altro tasto per continuare"
   ta$ = INPUT$(1): IF ta$ = CHR$(27) THEN EXIT DO
LOOP
```

```
END
SUB StampaArray (array%())
' Stampa un array di interi utilizzando virgole per la spaziatura.
   FOR k = LBOUND(array%) TO UBOUND(array%)
     PRINT array%(k),
   NEXT k
   PRINT : PRINT
END SUB

SUB BubbleSort (array%())
' Riordina un array di interi da cima a fondo
   cima = LBOUND(array%)
   fondo = UBOUND(array%)
   DO WHILE cima < fondo
     FOR qui = fondo TO cima + 1 STEP -1
       IF array%(qui) < array%(qui - 1) THEN
         SWAP array%(qui), array%(qui - 1)
       END IF
     NEXT qui
     cima = cima + 1
   LOOP
END SUB

SUB Mescola (array%())
' Mescola casualmente un array numerico.
' Assume che il limite inferiore sia uno (1).
   RANDOMIZE (TIMER)
   ' Trova la dimensione dell'array (numero di elementi)
   dimensione = UBOUND(array%)
   ' Scambia ogni elemento con uno selezionato casualmente

   FOR k = 1 TO dimensione
     indicecas = INT(dimensione * RND(1)) + 1
     SWAP array%(k), array%(indicecas)
   NEXT k

END SUB
```

5.5 Utilizzo di più procedure

I programmi QuickBASIC non sono limitati all'uso di una sola procedura, ma, come illustrato nel Programma 5.7, è possibile espandere un programma esistente; si potrebbe ad esempio avere un programma per tenere degli appunti con il seguente sottoprogramma che li stampa:

```
CALL Stamp(Note$())
  .
  .
  .
SUB Stamp (Elem$())
' Stampa un elemento dell'array (un appunto)
' alla volta da 1 a UBOUND(Elem$)
  CLS
  FOR num = 1 TO UBOUND(Elem$)
    PRINT Elem$(num)
    PRINT
  NEXT num
END SUB
```

Il Programma 5.8, STAMPA NOTE, si serve del sottoprogramma (Stamp) per inserire cinque brevi elementi di testo, che sono posti in un array chiamato Note$. Viene quindi aggiunto il sottoprogramma.

```
DECLARE SUB Stamp (Elem$())
REM ** STAMPA NOTE **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0508.BAS

' Inizializza e determina il numero di appunti
CLS : DEFINT A-Z
INPUT "Quante note "; num
DIM Note(1 TO num) AS STRING

' Prende gli appunti
FOR num = 1 TO UBOUND(Note$)
   PRINT num;
   LINE INPUT ">"; Note$(num)
   PRINT
NEXT num
' Stampa le note e termina
CALL Stamp(Note$())

END

SUB Stamp (Elem$())
' Stampa un elemento dell'array (un appunto)
' alla volta da 1 a UBOUND(Elem$)
   CLS
   FOR num = 1 TO UBOUND(Elem$)
     PRINT Elem$(num)
     PRINT
   NEXT num
 END SUB
```

Programma 5.8 STAMPA NOTE

Si digiti il programma principale del Programma 5.8 e lo si memorizzi con il comando Salva con nome del menu File, assegnandogli il nome UQB0508.BAS; si aggiunga quindi il sottoprogramma con il comando Nuova SUBroutine del menu Modifica e, per finire, si memorizzino insieme i due con il comando Salva del menu File. La Figura 5.7 illustra cinque appunti tipici stampati dal sottoprogramma dopo essere stati inseriti nella parte principale del programma.

```
Si possono passare gli elementi di un array ad un sottoprogramma come segue:
Il nome dell'array seguito da parentesi vuote ().
Il nome e le parentesi sono utilizzati nella lista degli argomenti.
Sono anche usati nella lista dei parametri del sottoprogramma.
LBOUND ed UBOUND sono utili per determinare la dimensione di un array.

Premere un tasto per continuare
```

Figura 5.7 Output del Programma 5.8

È possibile variare l'istruzione DIM in modo da incrementare o decrementare la dimensione dell'array senza dover modificare nulla nel programma, poiché LBOUND ed UBOUND si curano dei limiti superiore ed inferiore del ciclo FOR...NEXT che stampa le note.

È possibile aggiungere un secondo sottoprogramma al Programma 5.8: si ricorra al comando Apri programma del menu File per accedere al Programma 5.8, se non si trova già sul video, si sposti quindi il cursore verso il basso sino alla linea vuota sopra l'istruzione END, si digiti la linea che segue e si prema **Invio**:

```
CALL ContaCaratteri(Note$())
```

Questa istruzione passerà l'array Note$ al sottoprogramma ContaCaratteri; si acceda ora al comando Sub del menu Modifica, si digiti il nome del sottoprogramma come segue e si prema **Invio**:

```
Nome:  ContaCaratteri
Si digiti il seguente sottoprogramma:
 SUB ContaCaratteri (Rec$())
 ' Conta i caratteri negli appunti
   car = 0
   FOR num = 1 TO UBOUND(Rec$)
     car = car + LEN(Rec$(num))
   NEXT num
   PRINT "Il numero di caratteri nell'array e':"; car
 END SUB
```

Quando si è digitato il sottoprogramma si acceda al riquadro di comunicazione del comando SUBroutine del menu Visualizza, come illustrato nella Figura 5.8, ci si assicuri di aver evidenziato il nome del modulo principale e si prema **Invio**.

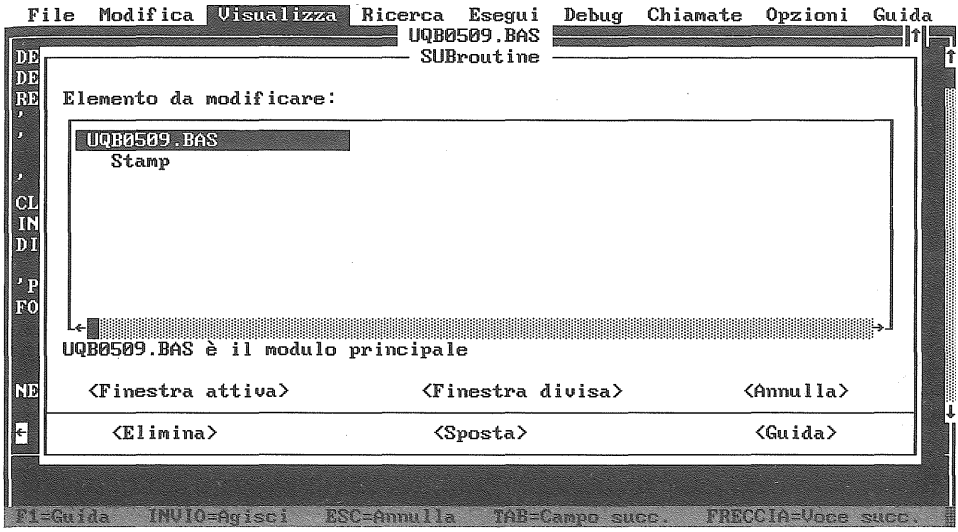


Figura 5.8 Riquadro di dialogo SUBroutine

Sul video comparirà il modulo principale: si modifichino il nome ed il numero del Programma in 5.9, STAMPA NOTE E CONTA CARATTERI, come segue:

```
REM ** STAMPA NOTE E CONTA CARATTERI **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File UQB0509.BAS
```

Si acceda ora a Salva con nome del menu File e si prema **Invio**; al prompt si digiti il nuovo nome di file.

Nome del file: UQB0509.BAS

Il sottoprogramma ContaCaratteri completa il Programma 5.9; la Figura 5.9 illustra un'esempio di esecuzione del programma utilizzando gli stessi cinque appunti illustrati nella Figura 5.7.

```
DECLARE SUB ContaCaratteri (Rec$())
DECLARE SUB Stamp (Elem$())
REM ** STAMPA NOTE E CONTA CARATTERI **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0509.BAS

' Inizializza e chiede il numero di appunti
CLS : DEFINT A-Z
INPUT "Quanti appunti "; num
```

```
DIM Note(1 TO num) AS STRING
'Prende gli appunti

FOR num = 1 TO UBOUND(Note$)
   PRINT num;
   LINE INPUT " "; Note$(num)
   PRINT
NEXT num
' Stampa gli appunti, conta i caratteri e termina
CALL Stamp(Note$())
CALL ContaCaratteri(Note$())
END

SUB ContaCaratteri (Rec$())
 ' Conta i caratteri negli appunti
   car = 0
   FOR num = 1 TO UBOUND(Rec$)
     car = car + LEN(Rec$(num))
   NEXT num
   PRINT "Il numero di caratteri nell'array e':"; car
END SUB

SUB Stamp (Elem$())
' Stampa un elemento dell'array (un appunto)
' alla volta da 1 ad UBOUND(Elem$)
   CLS
   FOR num = 1 TO UBOUND(Elem$)
     PRINT Elem$(num)
     PRINT
   NEXT num
END SUB
```

Programma 5.9 STAMPA NOTE E CONTA CARATTERI

```
Si possono passare gli elementi di un array ad un sottoprogramma come segue:

Il nome dell'array seguito da parentesi vuote ().

Il nome e le parentesi sono utilizzati nella lista degli argomenti.

Sono anche usati nella lista dei parametri del sottoprogramma.

LBOUND ed UBOUND sono utili per determinare la dimensione di un array.

Il numero di caratteri negli array e': 279


Premere un tasto per continuare
```

Figura 5.9 Output del Programma 5.9

5.6 Visualizzazione dei sottoprogrammi

Quando si richiama un programma QuickBASIC tutti i file contenenti i moduli che appartengono al programma vengono caricati nella memoria, ma solo il modulo principale viene visualizzato. Il menu Visualizza illustrato nella Figura 5.10 permette di aprire nuove finestre, visualizzare l'uscita dei programmi, osservare i moduli in memoria e personalizzare l'aspetto dello schermo.

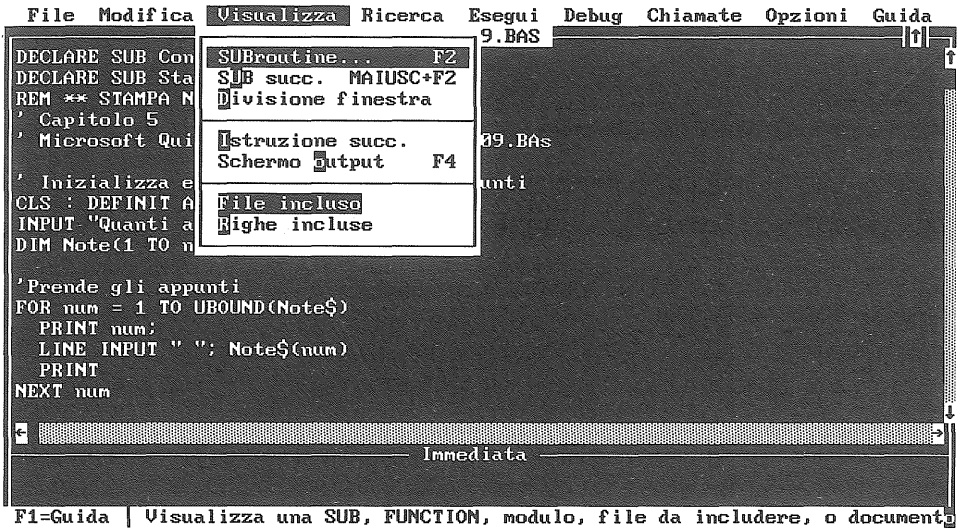


Figura 5.10 Menu Visualizza

Si evidenzi il comando Divisione finestra e si osservi il messaggio nella parte inferiore dello schermo: "Divide lo schermo in due finestre di visualizzazione". Si prema **Invio**, lo schermo viene suddiviso in due finestre con il cursore nella finestra superiore ed in ognuna è visualizzata una parte del modulo principale.

Si acceda nuovamente al menu Visualizza e, dopo aver evidenziato il comando SUBroutine, si prema **Invio**: il grande riquadro sullo schermo visualizza un elenco del modulo principale e dei sottoprogrammi che ora costituiscono il Programma 5.9, come nella Figura 5.11.

Il modulo principale, UQB5059.BAS, è evidenziato, ma è possibile evidenziare uno alla volta i sottoprogrammi o il modulo principale premendo i tasti Freccia Su e Freccia Giù. Il modulo evidenziato viene descritto sotto la finestra che elenca i moduli.

Si prema una volta il tasto **Freccia Giù**: verrà evidenziato il sottoprogramma ContaCaratteri, che è descritto sotto alla finestra con l'elenco: "CountCars è una SUB in

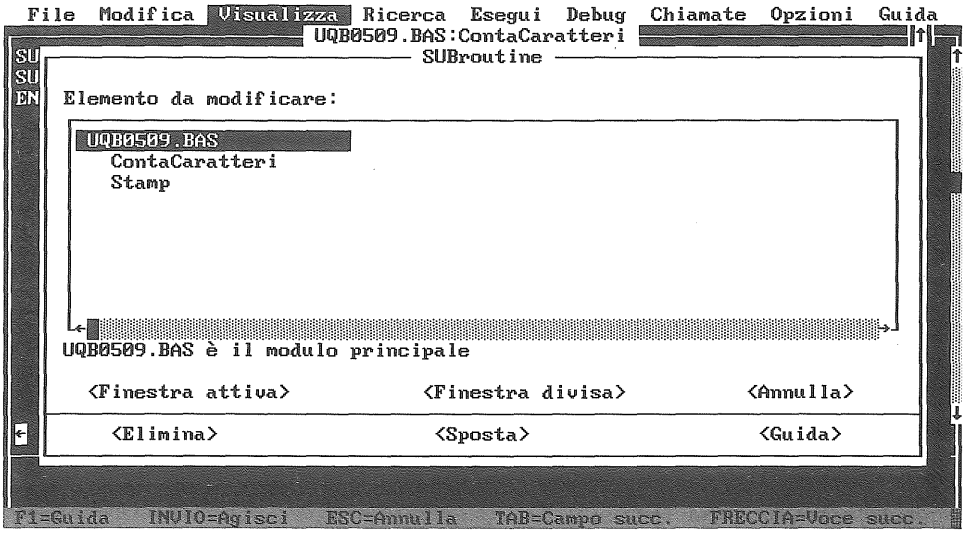


Figura 5.11 Riquadro di comunicazione SUBroutine

UQB0509.BAS". Si prema nuovamente il tasto **Freccia Giù** e verrà evidenziato Stamp: questo sottoprogramma è descritto sotto alla finestra con l'elenco "Stamp è una SUB in UQB0509.BAS". Si lasci evidenziata Stamp.

Si prema il tasto **Tab** due volte: il cursore si sposterà prima sul pulsante Finestra attiva e poi sul pulsante Finestra divisa. Si prema **Invio** mentre è attivo il pulsante Finestra divisa: così si tornerà alla finestra di visualizzazione dell'editor, che ora è suddivisa in due finestre, come illustrato nella Figura 5.12. Il sottoprogramma Stamp è nella finestra inferiore come anche il cursore poiché nel riquadro di comunicazione si era selezionato Stamp e Finestra divisa come finestra attiva. È ora possibile modificare a proprio piacere Stamp.

Si prema il tasto **F6**: il cursore si sposterà verso il basso alla piccola finestra nella parte inferiore dello schermo; quando il cursore è in questa finestra può essere digitato un comando immediato, come RUN. Si digiti **beep** e si prema **Invio**: si udirà immediatamente un suono.

Si prema nuovamente **F6**: il cursore si sposterà nella finestra di visualizzazione superiore, dove si trova il modulo principale; è ora possibile esaminare il modulo principale che, benché non sia tutto visualizzato nella finestra corta, può essere fatto scorrere con i tasti freccia.

Mentre il cursore è nella finestra in cui si trova il modulo principale, questo può anche essere modificato. Sullo schermo possono comparire contemporaneamente solamente due finestre di visualizzazione (oltre alla finestra Immediata) ed è possibile vedere due qualunque delle tre parti del programma: il modulo principale, SUB ContaCaratteri o SUB Stamp. Per variare la coppia di parti di programma nella finestra divisa, si acceda al comando SUB

```
File  Modifica  Visualizza  Ricerca  Esegui  Debug  Chiamate  Opzioni  Guida
                          UQB0509.BAS:Stamp
SUB Stamp (Elem$())
' Stampa un elemento dell'array (un appunto)
' alla volta da 1 ad UBOUND(Elem$)
  CLS
  FOR num = 1 TO UBOUND(Elem$)
    PRINT Elem$(num)
    PRINT
  NEXT num
                              UQB0509.BAS
DECLARE SUB ContaCaratteri (Rec$())
DECLARE SUB Stamp (Elems$())
REM ** STAMPA NOTE E CONTA CARATTERI **
' Capitolo 5
' Microsoft QuickBASIC 4.5  File: UQB0509.BAs

' Inizializza e chiede il numero di appunti
CLS : DEFINIT A - Z
INPUT "Quanti appunti "; num
                              Immediata
<MAIUSC+F1=Guida> <F6=Fin.> <F2=SUB> <F5=Esegui> <F8=Passo>      00001:002
```

Figura 5.12 Finestra di visualizzazione suddivisa

succ. del menu Visualizza e si prema **Invio**. Quando si è terminato il lavoro con la finestra divisa, si acceda al comando SUB succ. del menu Modifica e si prema **Invio**.

5.7 Sommario

Questo capitolo ha messo in rilievo la programmazione modulare con le procedure SUB...END SUB e FUNCTION...END FUNCTION. I sottoprogrammi e le funzioni di questo tipo sono procedure che si trovano al di fuori del programma principale: QuickBASIC organizza alfabeticamente questo livello più basso di codice e lo tiene separato dal resto del programma.

Le procedure vengono introdotte attraverso il menu Modifica e possono essere visionate con il menu Visualizza.

Sono stati introdotti gli array di QuickBASIC, trattando e provando in alcuni esempi i metodi per passare i dati degli array a procedure come sottoprogrammi e funzioni.

Capitolo 6

File sequenziali non strutturati

6.1 Tipi di file di dati
6.2 Attribuzione di un nome ad un file
6.3 Creazione di un file sequenziale
6.4 Lettura
6.5 Stampa
6.6 Aggiunte al file
6.7 Programma integrato per il file d'esempio
6.8 Sommario

Nei capitoli precedenti si sono utilizzati molti file; si sono ad esempio copiati file da dischetti MS-DOS e QuickBASIC per crearne uno di lavoro, si sono creati, memorizzati ed utilizzati file di programma contenenti i programmi QuickBASIC e moduli che possono essere inclusi in altri programmi. Questi programmi utilizzavano dati forniti da tastiera od inclusi nel programma come istruzioni di assegnamento (ad esempio, riga = 1) od in istruzioni DATA. Le istruzioni DATA sono memorizzate insieme al programma che si serve delle informazioni in esse contenute e sono un buon metodo per immagazzinare dati quando si lavora con una quantità limitata di informazioni. Quando invece è necessario manipolare grandi quantità di informazioni, le istruzioni DATA divengono scomode da introdurre, facilitano gli errori e occupano una quantità eccessiva di memoria del computer; un modo più efficiente per immagazzinare grandi quantità di informazioni è in file di dati su disco, che possono essere manipolati convenientemente da più programmi; inoltre un programma può servirsi di più di un file e si possono modificare i contenuti di un file di dati senza dover ricompilare il programma.

QuickBASIC può gestire due tipi di file di dati: file sequenziali e file ad accesso casuale; i file sequenziali vengono suddivisi in questo libro in file strutturati e non strutturati: in questo capitolo si apprenderà come creare ed utilizzare file sequenziali non strutturati. Il Capitolo 7 tratta i file sequenziali strutturati ed il Capitolo 8 i file ad accesso casuale.

6.1 Tipi di file di dati

Un file di dati è un insieme di record, generalmente memorizzato su un nastro magnetico o su un disco. Un file sequenziale ha un primo record, un secondo record, un terzo record e così via; i file sequenziali ricordano i tempi precedenti all'invenzione delle unità a dischi, quando i file di dati venivano registrati su nastri magnetici. Oggi il nastro magnetico viene utilizzato principalmente per archiviare e per fare copie di sicurezza dei file, mentre i file correnti vengono registrati su dischi. Anche su un disco i file sequenziali si comportano come se si trovassero su un nastro: per recuperare il record 7 il computer legge sequenzialmente il record 1, il record 2, il record 3 e così via sino a che non compare il record desiderato, numero 7. I file sequenziali hanno vantaggi e svantaggi.

Vantaggi

- I record possono essere di lunghezza qualunque, anche molto brevi o molto lunghi
- Poiché i record possono essere di lunghezza variabile, si ha un utilizzo efficiente dello spazio.
- Sono eccellenti per informazioni che non cambiano o che non cambiano frequentemente, ad esempio dati storici.

Svantaggi

- Poiché l'informazione è ricercata elemento per elemento, l'estrazione di informazioni da grandi file di dati è una operazione lunga.
- La modifica di un file sequenziale è scomoda, dal momento che è generalmente necessario riscrivere il file.
- Benché sia semplice aggiungere record al termine del file, per inserire dei record al suo interno è necessario riscrivere l'intero file ed anche per cancellare dei record è necessaria una completa riscrittura.
- Il riordinamento di un file sequenziale grande è notevolmente più complesso del riordinamento di un file ad accesso casuale.

Un file sequenziale è come la musica registrata su una cassetta: primo brano, secondo brano, terzo brano e così via; i brani i possono avere diversa lunghezza e per giungere al settimo è necessario percorrere tutto il nastro partendo dal primo e per cancellare od aggiungere un brano è necessario creare un nuovo nastro. Si pensi alla difficoltà nel creare un nastro in cui i brani compaiano tutti con un ordine completamente differente. In questo libro vengono presentati sia i file strutturati che non strutturati.

Non strutturati

Sono file di forma libera: ogni record è una stringa; un record può avere lunghezza pari a quella che può avere un stringa, molto lunga o molto corta; è possibile anche avere come record una stringa vuota "".

Strutturati

Un record è costituito da due o più parti, dette campi, ed ogni record ha gli stessi campi, che possono essere numerici o stringa. I campi stringa possono avere lunghezza pari a quella massima per una stringa (32767 caratteri); i campi numerici vengono immagazzinati in formato ASCII, un byte per cifra e sono limitati in lunghezza dalle variabili numeriche utilizzate per scrivere o leggere dai file.

Gran parte di questo capitolo è dedicata alla creazione ed all'uso di un file sequenziale non strutturato chiamato Notes, che è costituito da record in forma libera contenenti degli appunti. Un record può contenere qualsiasi cosa, ad esempio degli appunti su come utilizzare QuickBASIC, a condizione che questi siano digitati come un unica stringa di caratteri ASCII.

```
Primo record: "Questo e' il file Notes.DAT"
Secondo record: "Notes.DAT e' un file sequenziale non strutturato."
Terzo record: "Lo si utilizzi per gli appunti sull'uso di QuickBASIC."
Quarto record: "Ogni record e' una stringa, virgole e punto e virgola sono accettati."
Quinto record: "Chiamare Vanessa per chiarimenti sul Capitolo 7."
Sesto record: "Chiamare il dentista per un appuntamento urgente."
```

Come esempio di file strutturato, si pensi ad un programma che trova abbreviazioni di due lettere dei codici postali dei vari stati che compongono gli Stati Uniti: nel Capitolo 7 si creerà un file di dati con queste informazioni ed ogni record del file sarà costituito da due campi:

Record	Primo campo	Secondo campo
Primo Record	Alabama	AL
Secondo Record	Alaska	AK
Terzo Record	American Samoa	AS
.	.	.
.	.	.
.	.	.
Ultimo Record	Wyoming	WY

In questo file ogni record ha due campi: il primo campo (che sarà chiamato Stato$) ha lunghezza variabile ed il secondo campo (che sarà chiamato Abbr$) ha lunghezza fissa (due caratteri).

Un altro esempio di file strutturato è un catalogo di equipaggiamento da campeggio; in questo file ogni record è suddiviso in campi nel modo seguente.

Pag% Numero di pagina, un campo numerico.

NumCat$ Numero di catalogo, un campo stringa.

Descrizione$ Breve descrizione del prodotto, un campo stringa.

Prezzo Prezzo, campo numerico.

Grammi% Peso in grammi, un campo numerico.

Ed ora alcuni record:

Pag%	NumCat$	Descrizione$	Prezzo	Grammi%
5	33-972	Zaino	129.250	1824
10	47-865	Tenda	199.950	3157
19	50-336	Sacco a pelo	99.950	1653
25	40-027	Fornelletto	41.950	379
27	40-115	Kit per cucinare	29.950	884
31	45-820	Bussola	25.950	86
44	47-322	Coltello da campeggio	13.950	57

Si supponga ora che l'intero catalogo sia memorizzato su disco: sarebbe semplice scrivere un programma per esaminare e realizzare diverse configurazioni di equipaggiamento, complete di prezzo e peso totale. Poiché le informazioni del catalogo generalmente vengono modificati solo poche volte all'anno, questa sarebbe una applicazione ideale per un file sequenziale.

I file sequenziali vengono immagazzinati come file di testo in formato ASCII e quindi possono venire utilizzati da programmi di elaborazione testi o da altri programmi applicativi che accettano testo in formato ASCII: ci si potrebbe servire di uno di questi applicativi per vedere e modificare un file sequenziale.

Negli anni '50 l'introduzione del file su disco rese possibile l'uso di file ad accesso casuale, nei quali ogni record può essere recuperato direttamente e velocemente senza dover leggere nessun altro record. Se si desiderasse ad esempio il record 237, lo si potrebbe ottenere direttamente, senza leggere prima i precedenti 236 record. I file ad accesso casuale sono rigorosamente strutturati: tutti i record devono avere uguale lunghezza: è possibile impostarne la dimensione e se non la si specifica QuickBASIC assegna una dimensione di default di 128 caratteri.

I record possono essere suddivisi in campi, che devono avere dimensione fissa, cioè i campi corrispondenti dei vari record devono avere la stessa dimensione e devono essere disposti nello stesso ordine. I file ad accesso casuale hanno vantaggi e svantaggi.

Vantaggi

- Permettono l'accesso diretto a qualunque record.
- Sono facili da modificare: è possibile modificare qualunque record senza dover riscrivere il file.

- Rendono l'eliminazione o l'inserimento di un record più semplice rispetto ai file sequenziali.
- Sono più facili da riordinare rispetto ai file sequenziali.

Svantaggi

- Necessitano di una lunghezza fissa del record e del campo e quindi alcuni record o campi potrebbero essere parzialmente "vuoti".
- Richiedono un lavoro più lungo per l'impostazione del file: è necessario specificare la struttura e la dimensione dei campi.

Un file ad accesso casuale di stati ed abbreviazioni di due lettere conterrebbe 60 record per gli Stati Uniti, uno per ogni stato o territorio a cui è stata attribuita una abbreviazione in due lettere. Il nome più lungo sarebbe Federal States of Micronesia che è costituito da 30 caratteri e quindi ogni record dovrebbe contenere due campi.

Stato$ Nome dello stato o territorio, 30 caratteri.

Abbr$ Abbreviazione di due lettere, 2 caratteri.

Questo file permetterebbe l'accesso all'abbreviazione di due lettere di ogni stato, ma l'uso dello spazio sarebbe inefficiente; ad esempio, Iowa occuperebbe un campo di 30 caratteri.

Le specifiche relative alle informazioni che si intendono manipolare determinano il tipo di file di dati che deve essere utilizzato: si tenga presente la frequenza con cui i dati verranno modificati, aggiunti o cancellati. Generalmente i file sequenziali sono l'ideale per dati che non cambiano frequentemente, mentre i file ad accesso casuale sono la miglior scelta per dati che cambiano frequentemente e per informazioni alle quali si debba accedere rapidamente. Al termine di questo e dei prossimi due capitoli si dovrebbe essere in grado di scegliere ed impostare il tipo di file più adatto al proprio programma.

6.2 Attribuzione di un nome ad un file

Ai file sequenziali, come per i file MS-DOS, viene attribuito un nome, che deve essere conforme alle convenzioni del DOS sull'attribuzioni di nomi ai file, essi sono limitati ad otto caratteri più un'estensione di tre caratteri, come ad esempio Notes.Dat.

L'estensione di tre lettere (Dat) è stata scelta per indicare che si tratta di un file di dati.

I nomi di file possono essere composti dai seguenti caratteri:

- lettere dalla a alla z (minuscole o maiuscole);
- numeri da 0 a 9;
- caratteri speciali () { } @ # $ % ^ & ! - _ ' / "

In questo libro i nomi di file compaiono spesso in un misto di minuscolo e maiuscolo per renderli più leggibili, ma il DOS converte tutte le minuscole in maiuscole e quindi non si può ricorrere all'uso di maiuscole e minuscole per distinguere dei file.

Dal momento che il nome è essenziale per individuare un file, si rendano i nomi dei propri file i più descrittivi possibile entro il limite degli otto caratteri e, per tenere nota dei propri file, si potrebbe impostare un file separato che elenchi i nomi dei file e ne descriva i contenuti; a questo scopo ci si può servire di un file non strutturato. Questo file, che si potrebbe chiamare FileNomi.Dat, potrebbe avere l'aspetto seguente.

Record 1: Notes.Dat, contiene appunti di ogni tipo.

Record 2: StatiAbr.Dat, contiene i nomi degli stati e delle abbreviazioni.

Record 3: Cibo.Dat, contiene informazioni nutrizionali.

Record 4: Indirizz.Dat, contiene nomi, indirizzi e numeri di telefono.

Record 5: Camping.Cat, è un catalogo di equipaggiamento da campeggio.

Record 6: Prove.Dat, è per prove temporanee.

Dopo aver dato un nome al file, questo resterà immutato sino a che non lo si modifichi con il comando rename del DOS o con l'istruzione di QuickBASIC NAME. L'istruzione NAME ha la seguente forma:

```
NAME vecchio_nome_del_file AS nuovo_nome_del_file
```

dove vecchio_nome_del_file è il nome corrente del file e nuovo_nome_del_file è il nuovo nome che si vuole attribuire al file. Quando si rinomina un file non è possibile assegnargli una nuova designazione di unità a dischi; dopo l'uso dell'istruzione NAME il file rimane nello stesso spazio sul disco, ma con un nome differente.

6.3 Creazione di un file sequenziale

Un file di appunti è un semplice esempio di file sequenziale: si pensi ad ogni record come ad un cartoncino da 5 cm. per 12, in cui i record non sono strutturati in campi; il file è quindi in forma libera ed in ogni cartoncino è possibile avere un'appunto.

Per creare un file Notes.Dat si deve:

- aprire il file in modo da potervi introdurre le informazioni;
- introdurre da tastiera i record e porli nel file;
- chiudere il file dopo aver introdotto tutti i record.

Per portare a termine queste operazioni ci si servirà delle seguenti istruzioni di QuickBASIC:

OPEN	Per aprire un file.
LINE INPUT	Per ottenere un record da tastiera.
WRITE #	Per scrivere il record nel file.
CLOSE	Per chiudere un file.

Si crei dapprima il nuovo file e lo si apra per l'output in modo che si possano scrivere i record nel file. Nell'istruzione OPEN il file è chiamato Notes e gli è assegnato il numero di file 1.

```
OPEN "B:Notes.Dat" FOR OUTPUT AS #1
 drive^  ^nome del file            ^numero del file
```

Viene selezionata l'unità a dischetti B, ma, se si sta utilizzando un disco fisso od un'altra unità, si modifichi la lettera di specificazione nell'istruzione OPEN. Si osservi che Notes.Dat viene utilizzato come nome di file, conformemente alle convenzioni per i nomi di variabili ed etichette di linea: QuickBASIC riconosce Notes.Dat e NOTES.DAT come la stesso nome di file.

Se un file di nome Notes.Dat esistesse già sul disco , il modo di output illustrato nell' istruzione OPEN cancellerebbe tutti i contenuti di Notes.Dat prima di scrivere in esso dei nuovi dati. Se si desidera aggiungere dei nuovi dati al termine di un file esistente senza cancellare quanto già si trova in esso, ci si assicuri di utilizzare il modo append nell' istruzione OPEN, come descritto nel seguito di questo capitolo.

Prima di servirsi di un'istruzione OPEN...FOR OUTPUT, si controlli la directory del disco che viene utilizzato e ci si assicuri di utilizzare un nuovo nome di file quando si crea un nuovo file. Il modo append è sempre una sicura alternativa al modo output poiché permette anch'esso di ottenere un file creando un nuovo file con quel nome, se sul disco non è presente un file con nome Notes.Dat. L'istruzione LINE INPUT è adatta per introdurre record in un file sequenziale non strutturato: permette di introdurre fino a 255 caratteri, incluse le virgole; quando si preme **Invio**, l'informazione viene immagazzinata come una singola stringa (un record). L'istruzione LINE INPUT ha il seguente aspetto:

```
LINE INPUT "> "; record$
```

L'istruzione WRITE # viene utilizzata per scrivere una stringa come record in un file.

```
WRITE #1, record$
      ^numero del file
```

L'istruzione WRITE # inserisce delle virgole tra i campi mentre vengono scritti nel file, pone le stringhe tra virgolette ed infine pone un carattere di ritorno carello e di avanzamento linea dopo l'ultimo campo di un record. Nel prossimo capitolo si vedrà un altro metodo per scrivere informazioni in un file servendosi dell'istruzione PRINT #. Dopo aver introdotto tutti i record da tastiera ed averli scritti in un file lo si chiuda. L'istruzione che segue chiude il file numero 1, l'unico utilizzato nel programma nella prossima sezione per creare il file Notes.

```
CLOSE #1
```

6.3.1 Programma per la creazione del file di esempio

Il programma 6.1, CREA UN FILE, consiste di un programma principale ed un sottoprogramma; il programma principale è breve poiché svolge solamente due compiti: chiama il sottoprogramma (in cui viene creato il file) e termina l'esecuzione.

```
DECLARE SUB CreaFile
REM ** CREA UN FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0601.BAS

REM ** Imposta il tipo di default per le variabile ad intero **
DEFINT A-Z

REM ** Chiama il sottoprogramma di creazione del file e termina **
CALL CreaFile
END

SUB CreaFile STATIC
' Crea il file Notes.Dat. Ogni record e' una stringa.

  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Crea un nuovo file chiamato Notes.Dat."
  Text$(3) = "Porre il disco dati nel drive B e premere un tasto."
  Text$(4) = "Al prompt (>), digitare un record e premere Invio;"
  Text$(5) = "per finire il file premere Invio senza scrivere dati."
  CLS
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Attende che si prema un tasto per incominciare.
  LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
  untasto$ = INPUT(1)

  ' Definisce le righe da 1 a 20 come finestra per introdurre i record
  VIEW PRINT 1 TO 20
  CLS 2

  ' Apre il file NotePad.Dat sull'unita' B per l'output come file #1
  OPEN "B:NotePad.Dat" FOR OUTPUT AS #1

  ' Introduce i record da tastiera e scrive nel file
  DO
    LINE INPUT " "; record$
    PRINT
    IF record$ = "" THEN EXIT DO
```

```
    WRITE #1, record$
  LOOP
  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB
```

Programma 6.1 CREA UN FILE

Il sottoprogramma CreaFile crea un file di dati in forma libera per contenere i record che si introducono da tastiera: ogni record è limitato dal numero di caratteri accettati dall'istruzione LINE INPUT, cioè 255.

```
CREA UN FILE
REM ** CREA UN  FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0601.BAS

REM ** Imposta il tipo di default per le variabile ad intero **
DEFINT A-Z

REM ** Chiama il sottoprogramma di creazione del file e termina **
CALL CreaFile
END
```

Il sottoprogramma si serve della seguente istruzione VIEW PRINT per creare una finestra di testo, restringendo la zona nella quale compare l'output:

```
VIEW PRINT 1 TO 20
```

dove 1 è la prima linea della finestra e 20 l'ultima. L'informazione contenuta in una istruzione PRINT viene visualizzata in modo normale se questa è eseguita prima di un'istruzione VIEW PRINT. Il sottoprogramma CreaFile stampa le istruzioni per il suo uso prima dell'istruzione VIEW PRINT, le istruzione sono poste in una area fissa nella parte inferiore dello schermo, dalla riga 21 alla 25 e sono esterne all'area della finestra: rimangono fisse sullo schermo e non vengono interessate dalle linee stampate nella finestra, che è impostata dall'istruzione VIEW PRINT precedentemente data.

Dopo l'esecuzione dell'istruzione VIEW PRINT, l'istruzione CLS 2 pulisce il messaggio "Premere un tasto per iniziare" dalla finestra. Tutto l'output verso lo schermo compare nell'area definita da lineainiziale (1) e lineafinale (20) dell'istruzione VIEW PRINT.

Il sottoprogramma CreaFile apre il file Notes.Dat per l'output sull'unità B (se si sta utilizzando un'altra unità, si usi l'appropriata designazione). I record vengono introdotti uno alla volta in un ciclo DO...LOOP per mezzo di LINE INPUT e vengono scritti nel file dall'istruzione WRITE #. Se si preme **Invio** senza introdurre un record al prompt di LINE INPUT () si determina un'uscita dal ciclo.

```
  .
  .
OPEN "B:Notes.Dat" FOR OUTPUT AS #1
  .
DO
```

```
  LINE INPUT "> "; record$
  PRINT
  IF record$ = "" THEN EXIT DO
  WRITE #1, record$
LOOP
```

Quando si ha l'uscita da DO LOOP, il file viene chiuso, il sottoprogramma termina ed il controllo torna al programma principale.

```
  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB
```

6.3.2 Creazione del file

Per servirsi del programma di creazione di file sequenziali è necessario legare il programma principale al sottoprogramma: si introduca dapprima il programma principale, se lo si desidera lo si memorizzi su disco con il nome UQB0601.BAS e si introduca quindi, il sottoprogramma con i passi seguenti per mezzo del sistema di menu completo.

- Si acceda al menu Modifica e si evidenzi il comando SUB.
- Si prema **Invio**: viene visualizzata una finestra di dialogo, come illustrato nella Figura 6.1.

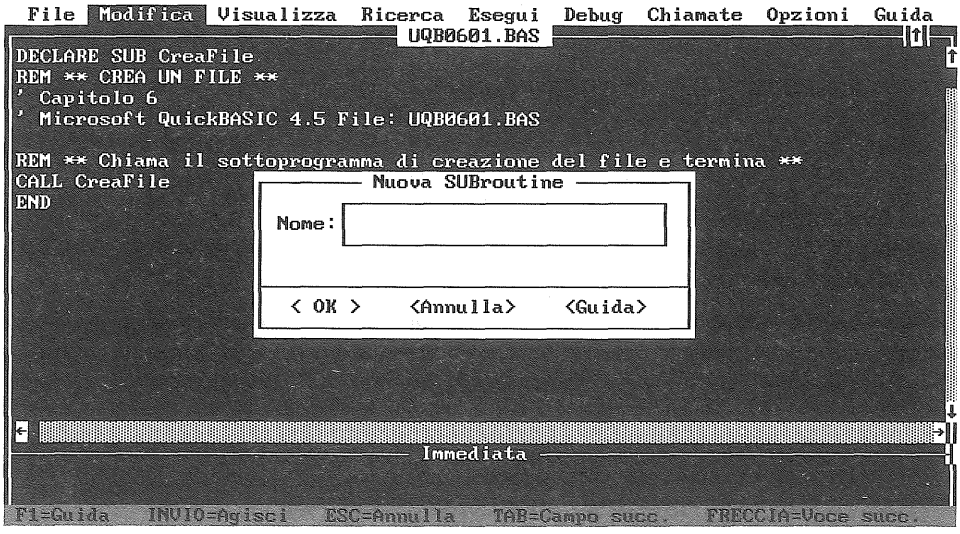


Figura 6.1 Riquadro di comunicazione Nuova SUBroutine

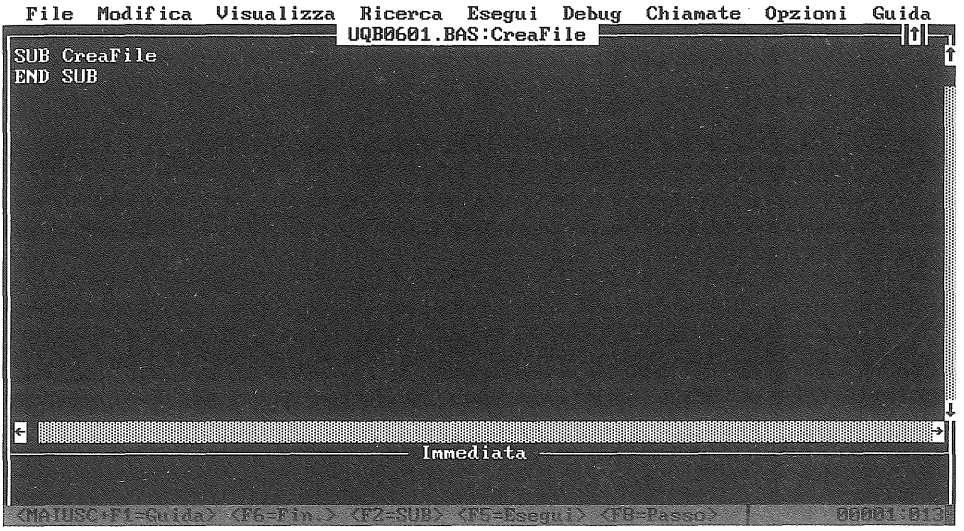


Figura 6.2 Schermo riportante le linee di apertura e chiusura del sottoprogramma

- Si digiti il nome del sottoprogramma CreaFile e si prema **Invio**; compare lo schermo illustrato nella Figura 6.2 e QuickBASIC pone automaticamente le linee di apertura e chiusura del sottoprogramma.
- Si termini di introdurre la prima linea del sottoprogramma:

```
SUB CreaFile STATIC
```

- Si introducano le altre linee del sottoprogramma:

```
SUB CreaFile STATIC
' Crea un file Notes.Dat. Ogni record e' una stringa

  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Crea un nuovo file chiamato Notes.Dat."
  Text$(3) = "Porre il disco dati nel drive B e premere un tasto."
  Text$(4) = "Al prompt (>), digitare un record e premere Invio;"
  Text$(5) = "per finire il file premere Invio senza scrivere dati."

  CLS

  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
       NEXT riga
       ' Attende che si prema un tasto per incominciare
       LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
```

```
        untasto$ = INPUT(1)
        ' Definisce le righe da 1 a 20 come finestra per introdurre i record
        VIEW PRINT 1 TO 20
    CLS 2

    ' Apre il file Notes.Dat sull'unita' B per l'output come file #1
    OPEN "B:Notes.Dat" FOR OUTPUT AS #1

    ' Introduce i record da tastiera e scrive nel file
    DO
      LINE INPUT " "; record$
      PRINT
      IF record$ = "" THEN EXIT DO
      WRITE #1, record$
    LOOP

    ' Chiude il file e termina il sottoprogramma
    CLOSE #1
  END SUB
```

- Dopo aver introdotto il sottoprogramma si acceda al menu File e si selezioni il comando Salva; il sottoprogramma CreaFile viene memorizzato con il programma principale come UQB0601.BAS.

Dopo aver memorizzato il programma, il sottoprogramma rimane sullo schermo. Si acceda al menu Visualizza e, mentre è evidenziato il comando SUBroutine, si prema **Invio**. Compare il riquadro di comunicazione illustrato nella Figura 6.3.

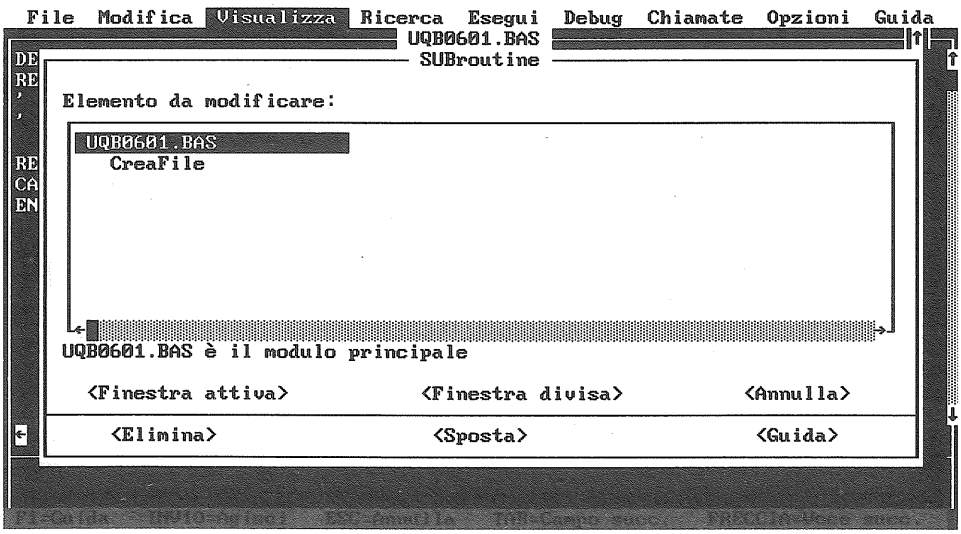


Figura 6.3 Riquadro di comunicazione SUBroutine del menu Visualizza

Dal riquadro di comunicazione è possibile scegliere se si desidera vedere e modificare il programma principale, UQB0601.BAS, o il sottoprogramma, CreaFile. Mentre è evidenziato UQB0601.BAS si prema **Invio**: QuickBASIC torna alla finestra di visualizzazione dopo avervi posto il programma principale. Si osservi la prima linea del programma principale, QuickBASIC ha automaticamente aggiunto la linea:

```
DECLARE SUB CreaFile ()
```

L'istruzione DECLARE permette ai programmi di chiamare procedure che vengono definite più tardi.

Il programma principale ed il sottoprogramma sono ora pronti per essere utilizzati: si acceda al menu Esegui e, con il comando Avvia evidenziato, si prema **Invio**: le istruzioni sono visualizzate come viene illustrato nella Figura 6.4.

```
Premere un tasto per iniziare


Crea un nuovo file chiamato Notes.Dat.
Porre il disco dati nel drive B e premere un tasto.
Al prompt (>), digitare un record e premere Invio;
per finire il file premere Invio senza scrivere dati.
```

Figura 6.4 Schermata di apertura del sottoprogramma CreaFile

Si digitino gli appunti mostrati nella Figura 6.5: si prema **Invio** solamente dopo aver completato ognuno dei record.

```
I nomi dei file nelle istruzioni OPEN sono espressioni stringa.
Le istruzioni OPEN possono contenere le lettere a-z, le cifre 0-9 ed alcuni caratteri speciali.
Alcuni caratteri speciali sono () {} @ # $ % ^ & ! - _ ' / ".
L'istruzione OPEN puo' anche contenere la specifica di un'unita' disco.
I programmi possono utilizzare file di un'altra unita'.
Per creare un file si utilizza l'istruzione OPEN con FOR OUTPUT.
Per aggiungere record ad un file si utilizza APPEND nell'istruzione OPEN.
APPEND aggiunge record alla fine del file.
Per scrivere record in un file aperto si utilizza WRITE #.
WRITE # racchiude i record tra virgolette.
Per introdurre una stringa contenente virgole si utilizza LINE INPUT #.
INPUT # legge i caratteri di un file sequenziale.
Per chiudere un file da programma si utilizza l'istruzione CLOSE.
CLOSE rilascia il numero del file che puo' essere utilizzato in un'altra OPEN.
```

Figura 6.5 Note da digitare in CreaFile

Dopo aver introdotto gli appunti della Figura 6.5, si prema il tasto **Invio** all'ultimo prompt senza introdurre testo: il sottoprogramma abbandona il DO...LOOP e chiude il file, il

controllo torna al modulo principale del programma e l'esecuzione termina con un'istruzione END.

6.4 Lettura

Ci si serva ancora una volta di un breve programma principale per accedere ad un nuovo sottoprogramma chiamato EsaminaFile: il programma principale chiama il sottoprogramma e termina l'esecuzione. Si digiti il programma principale che segue e lo si memorizzi sullo stesso disco del Programma 6.1 con il nome UQB0602.BAS.

```
REM ** ESAMINA UN FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0602.BAS

REM ** Imposta il tipo di default per le variabili ad intero **
DEFINT A-Z
REM ** CHIAMA Esamina File, poi termina **
CALL EsaminaFile
END
```

Il sottoprogramma EsaminaFile stampa le istruzioni nelle linee da 21 a 25 sullo schermo ed imposta una finestra (linee da 1 a 20) per la scansione del file, quindi apre Notes.Dat (il file di dati precedentemente creato) per l'input come file numero 1.

```
OPEN "B:Notes.Dat" FOR INPUT AS #1
```

Viene utilizzata LINE INPUT # in un ciclo DO...LOOP per leggere dal file un record alla volta ed i record sono stampati sullo schermo uno alla volta. Poiché ci si era serviti dell'istruzione WRITE # per creare il file, le virgolette al principio ed alla fine del record vengono stampate come parte di ogni record. Per passare al record successivo si prema qualunque tasto eccetto U. Il ciclo DO...LOOP dispone di due possibili uscite: premendo U si forza l'uscita o, quando sono stati letti tutti i record e si preme un tasto diverso da U, viene incontrato l'indicatore di fine del file (EOF) che causa l'uscita dall'istruzione DO UNTIL EOF=(1) al principio del ciclo.

```
DO UNTIL EOF(1)
  LINE INPUT #1, record$
  PRINT record$
  PRINT
  tasto$ = INPUT$(1)
  IF UCASE$(tasto$) = "U" THEN EXIT DO
LOOP
```

Quando si opera un'uscita dal ciclo, il file viene chiuso, il sottoprogramma termina ed il controllo torna al programma principale, che termina anch'esso.

6.4.1 Programma di lettura di file sequenziali

Se non si è ancora introdotto il programma principale, lo si faccia ora, mentre, se lo si è fatto e questo non compare nella finestra di visualizzazione, si acceda ad esso con il comando Apri programma del menu File. Quando il programma principale si trova nella finestra di visualizzazione si è pronti per introdurre il sottoprogramma.

Si selezioni il comando nuova SUBroutine del menu Modifica, si prema **Invio** e comparirà il riquadro di comunicazione illustrato nella Figura 6.1. Si digiti EsaminaFile e si prema **Invio**: comparirà uno schermo quasi completamente vuoto, con le linee iniziale e finale del nuovo sottoprogramma nella parte superiore.

```
SUB EsaminaFile_
END SUB
```

Si digiti il resto del sottoprogramma EsaminaFile, come illustrato più sotto e, quando si è terminato, si richiami il comando Salva del menu File per memorizzare il sottoprogramma insieme al programma principale.

```
SUB EsaminaFile STATIC
' Percorre il file Notes.Dat, un record alla volta

  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Esamina il file Notes un record alla volta."
  Text$(3) = "Porre il disco dati nel drive B e premere un tasto:"
  Text$(4) = "verra' visualizzato il primo record nella finestra."
  Text$(5) = "Premere la barra spaziatrice per i successivi od U per uscire."
  CLS
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Definisce le righe da 1 to 20 come finestra per esaminare i record
  VIEW PRINT 1 TO 20

  ' Attende che venga premuto un tasto per incominciare
  LOCATE 10, 28:PRINT "Premere un tasto per iniziare"
  untasto$ = INPUT$(1)
  CLS 2                          'Pulisce solamente la finestra

  ' Apre il file Notes.Dat sull'unita' B per l'input come file 1
  OPEN "B:Notes.Dat" FOR INPUT AS #1

  ' Legge un record quando viene premuto un tasto diverso da U
  DO UNTIL EOF(1)
    LINE INPUT #1, record$
    PRINT record$
```

```
    PRINT
    tasto$ = INPUT$(1)
    IF UCASE$(tasto$) = "U" THEN EXIT DO
  LOOP

  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB
```

Questo completa il Programma 6.2, ESAMINA UN FILE.

Dopo aver memorizzato il programma, si acceda al comando Avvia del menu Esegui e si prema **Invio**: le istruzioni sono visualizzate insieme al prompt "Premere un tasto per iniziare". Si prema un tasto qualunque per visualizzare sullo schermo il primo record, come illustrato nella Figura 6.6.

Si ponga attenzione alle virgolette: quando viene utilizzata l'istruzione LINE INPUT # per leggere un record da un file, le virgolette vengono incluse come parte del record.

```
                Premere un tasto per iniziare



    Esamina il file Notes un record alla volta.

    Porre il disco dati nel drive B e premere un tasto:

    verra' visualizzato il primo record nella finestra.

    Premere la barra spaziatrice per i successivi od U per uscire.
```

Figura 6.6 Schermo di apertura della subroutine EsaminaFile

Ogni volta che si preme un tasto diverso da U, un nuovo record viene visualizzato sul video; quando si sono osservati tutti i record desiderati, si prema il tasto U per terminare, oppure si proceda sino a che non si siano letti tutti i record; la condizione di fine del file (EOF) concluderà il ciclo DO...LOOP.

```
DECLARE SUB EsaminaFile ()
REM ** ESAMINA UN FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0602.BAS

REM ** Imposta il tipo di default per le variabili ad intero **
DEFINT A-Z

REM ** CHIAMA Esamina File, poi termina **
CALL EsaminaFile
END

SUB EsaminaFile STATIC
```

```
' Percorre il file Notes.Dat, un record alla volta
  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Esamina il file Notes un record alla volta."
  Text$(3) = "Porre il disco dati nel drive B e premere un tasto:"
  Text$(4) = "verra' visualizzato il primo record nella finestra."
  Text$(5) = "Premere la barra spaziatrice per i successivi od U per uscire."
  CLS
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Definisce le righe da 1 to 20 come finestra per esaminare i record
  VIEW PRINT 1 TO 20

  ' Attende che venga premuto un tasto per incominciare
  LOCATE 10, 28:PRINT "Premere un tasto per iniziare"
  untasto$ = INPUT$(1)
  CLS 2                          'Pulisce solamente la finestra

  ' Apre il file Notes.Dat sull'unita' B per l'input come file 1
  OPEN "B:Notes.Dat" FOR INPUT AS #1
  ' Legge un record quando viene premuto un tasto diverso da U
  DO UNTIL EOF(1)
    LINE INPUT #1, record$
    PRINT record$
    PRINT
    tasto$ = INPUT$(1)
    IF UCASE$(tasto$) = "U" THEN EXIT DO
  LOOP
  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB
```

Programma 6.2 ESAMINA UN FILE

Si hanno ora a disposizione i programmi per creare e leggere un file sequenziale di dati non strutturato. Come prossimo passo sarà necessario un programma per inviare i record alla stampante in modo da crearne una copia su carta.

6.5 Stampa

Il programma principale volto alla stampa dei record del file Notes è molto simile a quelli dei primi due programmi di questo capitolo. Si introduca il programma principale come segue e lo si memorizzi con il nome UQB0603.BAS:

```
REM ** STAMPA UN FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0603.BAS

REM ** Imposta il tipo di default per le variabili ad intero **
DEFINT A-Z

REM ** Chiama la subroutine di stampa e termina **
CALL StampaFile
END
```

6.5.1 Programma di stampa di file sequenziali

Il sottoprogramma StampaFile non si ferma in attesa che si prema un tasto dopo la stampa di ogni record, ma continua a stampare fino a che non sia stata raggiunto il termine del file, oppure fino a che non venga premuto il tasto U per interrompere. Si introduca il sottoprogramma come in precedenza e lo si memorizzi insieme al programma principale sullo stesso disco dei file di dati precedentemente memorizzati.

Si analizzi il Programma 6.3, STAMPA UN FILE e si osservino le somiglianze e differenze tra quest'ultimo ed il Programma 6.2: viene utilizzata un'istruzione INPUT # anziché LINE INPUT # e, quando viene utilizzata un'istruzione INPUT #, il record viene stampato senza le virgolette di apertura e chiusura.

```
DECLARE SUB StampaFile ()
REM ** STAMPA UN FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0603.BAS

REM ** Imposta il tipo di default per le variabili ad intero **
DEFINT A-Z

REM ** Chiama la subroutine di stampa e termina **
CALL StampaFile
END

SUB StampaFile STATIC
' Stampa il file Notes.Dat

  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Invia il file Notes alla stampante."
  Text$(3) = "Porre il disco dei dati nel drive B."
  Text$(4) = "Assicurarsi che la stampante sia pronta e premere un tasto."
  Text$(5) = "Per interrompere premere il tasto U."
  CLS
  FOR riga = 1 TO 5
```

```
  LOCATE riga + 20, 3: PRINT Text$(riga);
NEXT riga

' Definisce le linee da 1 a 20 come finestra
' dove visualizzare i record durante la stampa
VIEW PRINT 1 TO 20

' Attende la pressione di un tasto per cominciare.
LOCATE 10, 28:PRINT "Premere un tasto per iniziare"
untasto$ = INPUT$(1)
CLS 2                          'Cancella solo la finestra

' Apre il file Notes.Dat in input come file numero 1
OPEN "B:Notes.Dat" FOR INPUT AS #1
' Legge i record, li visualizza e li stampa
DO UNTIL EOF(1)
  tas$ = INKEY$: IF UCASE$(tas$) = "U" THEN EXIT DO
  INPUT #1, record$
  PRINT record$
  LPRINT record$
  LPRINT
LOOP

' Chiude il file e termina il sottoprogramma
CLOSE #1
END SUB
```

Programma 6.3 STAMPA UN FILE

Ci si assicuri che la stampante sia accesa e pronta alla stampa prima di eseguire il programma.

Quando la stampante è pronta, si acceda al comando Avvia del menu Esegui e si prema **Invio**: si noti che i record vengono inviati tanto allo schermo quanto alla stampante. Il file viene stampato come illustrato nella Figura 6.7.

```
I nomi dei file nelle istruzioni OPEN sono espressioni stringa.
Le istruzioni OPEN possono contenere le lettere a-z, le cifre 0-9 ed
alcuni caratteri speciali.
Alcuni caratteri speciali sono () {} @ # $ % ^ & ! - _ ' / ".
L'istruzione OPEN puo' anche contenere la specifica di un'unita' disco.
I programmi possono utilizzare file di un'altra unita'.
Per creare un file si utilizza l'istruzione OPEN con FOR OUTPUT.
Per aggiungere record ad un file si utilizza APPEND nell'istruzione OPEN.
APPEND aggiunge record alla fine del file.
Per scrivere record in un file aperto si utilizza WRITE #.
```

```
WRITE # racchiude i record tra virgolette.
Per introdurre una stringa contenente virgole si utilizza LINE INPUT #.
INPUT # legge i caratteri di un file sequenziale.
Per chiudere un file da programma si utilizza l'istruzione CLOSE.
CLOSE rilascia il numero del file che puo' essere utilizzato in un'altra OPEN.
```

Figura 6.7 Il file Notes.Dat stampato dalla subroutine StampaFile

6.6 Aggiunte al file

Proseguendo nella lettura di questo libro probabilmente si desidererà aggiungere dei nuovi appunti al file Notes: è possibile farlo utilizzando il modo append nell'istruzione OPEN.

```
OPEN "B:Notes.Dat" FOR APPEND AS #1
```

Il modo append posiziona automaticamente il puntatore del file al termine del file Notes in modo che i record introdotti vengano aggiunti alla fine del file.

```
Primo record   Appunto 1
.
.
.
Ultimo record  Appunto n
Nuovo record
        ^puntatore di file al successivo record disponibile
```

Il programma principale è simile ai precedenti ed il sottoprogramma AggiungiFile è simile al sottoprogramma CreaFile del Programma 6.1: per creare un nuovo file ci si può infatti servire di uno qualunque dei due programmi.

6.6.1 Programma di aggiunta al file sequenziale

Si introduca solamente il programma principale del Programma 6.4, AGGIUNGE AD UN FILE, lo si memorizzi con il nome UQB0604.BAS e quindi, con il comando SUBroutine del menu Modifica, si introduca il sottoprogramma.

Si dia a quest'ultimo il nome AggiungiFile e lo si memorizzi insieme al programma principale per mezzo del comando Salva del menu File.

Si mandi in esecuzione il programma con il comando Avvia del menu Esegui: le istruzioni vengono stampate nella parte inferiore dello schermo e nella parte centrale della finestra il messaggio "Premere un tasto per iniziare". Quando si preme un tasto, la finestra viene pulita e viene visualizzata una freccia a destra in attesa di un nuovo record: si digitino gli appunti desiderati ed al termine si prema **Invio** senza premere alcun tasto.

Una volta completato il Programma 6.4 si dispone dei seguenti programmi volti alla manipolazione del testo in file sequenziali non strutturati:

Programma 6.1 CREA UN FILE

Programma 6.2 ESAMINA UN FILE

Programma 6.3 STAMPA UN FILE

Programma 6.4 AGGIUNGE AD UN FILE

```
DECLARE SUB AggiungiFile ()
REM ** APPENDE AD UN FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0604.BAS

REM ** Imposta le variabili di default ad interi **
DEFINT A-Z

REM ** Chiama la subroutine AggiungiFile SUB e termina **
CALL AggiungiFile
END

SUB AggiungiFile STATIC
' Aggiunge informazioni al file Notes.Dat
   ' Pone le istruzioni nelle linee da 21 a 25
   Text$(1) = STRING$(74, 196)
   Text$(2) = "Aggiunge record al file Notes."
   Text$(3) = "Porre il disco dei dati nel drive B e premere un tasto."
   Text$(4) = "Al prompt (>), digitare un record e premere Invio."
   Text$(5) = "Premere Invio senza digitare alcun dato per terminare."
   CLS
   FOR riga = 1 TO 5
     LOCATE riga + 20, 3: PRINT Text$(riga);
   NEXT riga
   ' Definisce le righe da 1 to 20 come finestra per i record da aggiungere
   VIEW PRINT 1 TO 20
   ' Attende la pressione di un tasto per iniziare
   LOCATE 10, 28:PRINT "Premere un tasto per iniziare."
   untasto$ = INPUT$(1)
   CLS 2                          'Cancella solo la finestra
```

```
  ' Apre il file Notes.Dat nel drive B per aggiungere come file numero 1
  OPEN "B:Notes.Dat" FOR APPEND AS #1
  ' Legge i record da tastiera e li aggiunge al file
  DO
    LINE INPUT "> "; record$
    PRINT
    IF record$ = "" THEN EXIT DO
    WRITE #1, record$
  LOOP

  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB
```

Programma 6.4 APPENDE AD UN FILE

Ogni programma assolve ad un compito distinto, ma i programmi sono assimilati dal fatto di operare tutti con lo stesso file su disco, Notes.Dat, e sarebbe quindi opportuno unirli in un unico programma che permetta di scegliere quale funzione eseguire, come descritto nel seguito.

6.7 Programma integrato per il file d'esempio

Nei capitoli precedenti si è appreso come servirsi di di molti strumenti: combinando i quattro programmi sviluppati in questo capitolo, si avrà l'opportunità di esercitare le tecniche di editing apprese nel Capitolo 4. Per creare un programma integrato sono necessari un programma principale di coordinamento ed i quattro sottoprogrammi utilizzati nei Programmi 6.1, 6.2, 6.3 e 6.4.

Ognuno di questi programmi è costituito da un breve programma principale che chiama il sottoprogramma necessario ed il programma principale per il programma integrato sarà leggermente più lungo. È necessario operare una selezione da un elenco di funzioni che il programma può eseguire ed un menu è un buon metodo per visualizzare le selezioni: le voci sono:

Crea un file

Esamina un file

Stampa un file

Aggiunge ad un file

Uscita

Il semplice menu con cinque voci può essere visualizzato con le seguenti stringhe:

```
Menu$(1) = "Crea un file"
Menu$(2) = "Esamina un file"
Menu$(3) = "Stampa un file"
Menu$(4) = "Aggiunge ad un file"
Menu$(5) = "Uscita"
```

È possibile stampare il testo e tracciare un semplice bordo intorno alle scelte o, se si desidera, è possibile aggiungere del colore od anche della grafica.

Un breve messaggio dovrebbe illustrare come utilizzare il menu.

```
Menu$(6) = "Digitare la prima lettera dell'opzione scelta (C, E, S, A od U)."
```

La lettera selezionata dovrebbe chiamare il corretto sottoprogramma e la struttura di QuickBASIC SELECT CASE è l'ideale per questo compito.

```
  ' Scelta
  lettera$ = UCASE$(INPUT(1))
  SELECT CASE lettera$
    CASE "C"
      CALL CreaFile
    CASE "E"
      CALL EsaminaFile
    CASE "S"
      CALL StampaFile
    CASE "A"
      CALL AggiungiFile
    CASE "U"
      END
    CASE ELSE
      LOCATE 21, 5: PRINT "Non e' stata premuto C, E, S, A od U."
      LOCATE 22, 5: PRINT "Premere un tasto per ritentare."
      ripetiz$ = INPUT$(1)
  END SELECT
```

Il menu e SELECT CASE costituiscono gran parte del programma principale, qui illustrato:

```
REM ** Visualizza il menu e chiede una scelta **
DO
  ' Scelte possibili
  Menu$(1) = "Crea un file"
  Menu$(2) = "Esamina un file"
  Menu$(3) = "Stampa un file"
  Menu$(4) = "Aggiunge ad un file"
  Menu$(5) = "Uscita"
  Menu$(6) = "Digitare la prima lettera dell'opzione scelta (C, E, S, A od U)."
  ' Visualizza il menu
  VIEW PRINT 1 TO 25: CLS
  FOR riga = 1 TO 5
    LOCATE riga + 5, 25: PRINT Menu$(riga)
```

```
  NEXT riga
  LOCATE 21, 5: PRINT Menu$(6)
  LOCATE 4, 20: PRINT CHR$(218); STRING$(29, 196); CHR$(191);
  LOCATE 12, 20: PRINT CHR$(192); STRING$(29, 196); CHR$(217);
  FOR riga = 5 TO 11
    LOCATE riga, 20: PRINT CHR$(179);
    LOCATE riga, 50: PRINT CHR$(179);
  NEXT riga
    ' Scelta
    lettera$ = UCASE$(INPUT(1))
    SELECT CASE lettera$
      CASE "C"
        CALL CreaFile
      CASE "E"
        CALL EsaminaFile
      CASE "S"
        CALL StampaFile
      CASE "A"
        CALL AggiungiFile
      CASE "U"
        END
      CASE ELSE
        LOCATE 21, 5: PRINT "Non e' stata premuto C, E, S, A od U."
        LOCATE 22, 5: PRINT "Premere un tasto per ritentare."
        ripetiz$ = INPUT$(1)
    END SELECT
    CLS2
LOOP
END
```

Lo si digiti e lo si memorizzi con il nome UQB0605.BAS.

6.7.1 Assemblaggio delle singole parti

Dopo aver memorizzato il programma sarà necessario includere i precedenti programmi (UQB0601.BAS,UQB0602.BAS,UQB0603.BAS e UQB0604.BAS) ed eseguire alcune operazioni di modifica. I sottoprogrammi CreaFile, EsaminaFile, StampaFile ed AggiungiFile sono necessari per il programma integrato, ma i programmi principali ad essi legati devono essere cancellati. Le operazioni da eseguire per ognuno dei quattro programmi sono:

- includere il programma nel nuovo programma principale;
- spostare l'istruzione DECLARE del programma incluso al principio del programma principale integrato;
- cancellare il breve programma che chiamava il sottoprogramma.

6.7.2 Inclusione di CREA UN FILE in NOTES CON MENU

Con il programma principale di UQB0605.BAS nella finestra di visualizzazione, si sposti il cursore sotto all'ultima linea del programma.

```
        .
        .
        .
    END SELECT
    CLS 2
  LOOP
  END
  _
  ^ cursore
```

Questo è il punto in cui UQB0601.BAS verrà incluso in UQB0605.BAS. Si acceda al comando Merge del menu File e si prema **Invio**; compare un riquadro di comunicazione, come illustrato nella Figura 6.8. Si prema TAB per spostare il cursore alla casella File, si prema il tasto **Freccia in basso** per evidenziare il nome di file UQB0601.BAS e quindi si prema **Invio**: UQB0601.BAS verrà incluso al termine del programma principale,

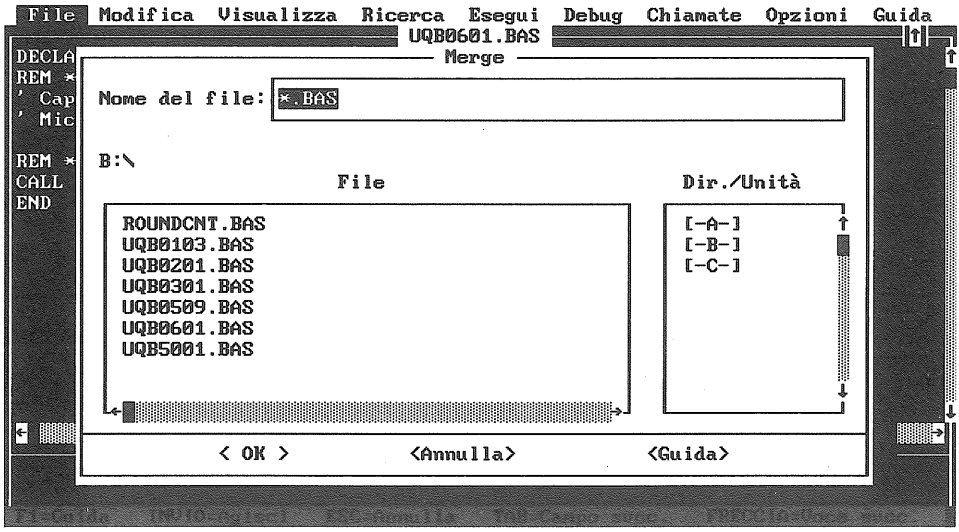


Figura 6.8 Riquadro di comunicazione di Merge

UQB0601.BAS. Ora è giunto il momento di applicare le tecniche di editing apprese nel Capitolo 4: si osservi che l'istruzione DECLARE per il sottoprogramma si trova sotto al programma principale.

```
  END SELECT
  CLS 2
LOOP
END
DECLARE SUB CreaFile ()
```

^istruzione DECLARE in posizione errata

.
.
.

L'istruzione DECLARE deve essere spostata al principio del file. Il comando Taglia del menu Modifica porta il testo selezionato nel ritaglio, lo si utilizzerà per tagliare l'istruzione DECLARE compiendo i seguenti passi:

- Si porti il cursore alla lettera D dell'istruzione DECLARE e si prema **Shift+Freccia Giù** per selezionare l'istruzione:

```
DECLARE SUB CreaFile ()
```

- Con il comando evidenziato, si acceda al menu Modifica, si selezioni il comando Taglia e si prema **Invio**: la linea sparisce immediatamente dallo schermo per passare al ritaglio, dove non può essere vista.
- Si prema **Ctrl+Iniz** per spostarsi al principio del file (la prima linea del programma principale).

```
REM ** NOTES CON MENU **
' Capitolo 6
  .
  .
  .
```

- Si acceda al comando Incolla del menu Modifica e si prema **Invio**: l'istruzione DECLARE verrà portata dal ritaglio alla prima linea del programma principale.

```
DECLARE SUB CreaFile ()
REM ** NOTES CON MENU **
' Capitolo 6
```

Si cancelli ora il programma principale di UQB0601.BAS incluso: si visualizzi il principio del programma incluso con il tasto **PgAv** e con i tasti freccia si porti il cursore sotto alla R di REM ** CREA UN FILE **, quindi si prema **Shift+Freccia Giù** fino a che non siano evidenziate le linee seguenti.

```
REM ** CREA UN FILE **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0601.BAS

REM ** Imposta il tipo di default per le variabile ad intero **
DEFINT A-Z

REM ** Chiama il sottoprogramma CreaFile e termina **
CALL CreaFile
END
```

Si acceda nuovamente al menu Modifica, si selezioni il comando Cancella e si prema **Invio**. Le linee scompaiono dallo schermo e questa volta non sono finite nel ritaglio, ma sono perse. Si ripeta il procedimento per ognuno dei tre programmi: UQB0602.BAS, UQB0603.BAS e UQB0604.BAS:

- includere il programma;
- spostare l'istruzione DECLARE al principio del file;
- cancellare il resto del file incluso.

Quando si sono terminate queste operazioni per ognuno dei programmi si avrà a disposizione il Programma 6.5, NOTES CON MENU, completo dei quattro sottoprogrammi: lo si memorizzi con il comando Salva del menu File.

```
DECLARE SUB AggiungiFile ()
DECLARE SUB StampaFile ()
DECLARE SUB EsaminaFile ()
DECLARE SUB CreaFile ()
REM ** NOTES CON MENU **
' Capitolo 6
' Microsoft QuickBASIC 4.5  File: UQB0605.BAS

REM ** Imposta il tipo di default per le variabile ad intero **
DEFINT A-Z
REM ** Visualizza il menu e chiede una scelta **
DO
  ' Scelte possibili
  Menu$(1) = "Crea un file"
```

```
  Menu$(2) = "Esamina un file"
  Menu$(3) = "Stampa un file"
  Menu$(4) = "Aggiunge ad un file"
  Menu$(5) = "Uscita"
  Menu$(6) = "Digitare la prima lettera dell'opzione scelta (C, E, S, A od U)."

  ' Visualizza il menu
  VIEW PRINT 1 TO 25: CLS
  FOR riga = 1 TO 5
    LOCATE riga +5, 25: PRINT Menu$(riga)
  NEXT riga
  LOCATE 21, 5: PRINT Menu$(6)
  LOCATE 4, 20: PRINT CHR$(218); STRING$(29, 196); CHR$(191);
  LOCATE 12, 20: PRINT CHR$(192); STRING$(29, 196); CHR$(217);
  FOR riga = 5 TO 11
    LOCATE riga, 20: PRINT CHR$(179);
    LOCATE riga, 50: PRINT CHR$(179);
  NEXT riga
    ' Scelta
    lettera$ = UCASE$(INPUT(1))
    SELECT CASE lettera$
      CASE "C"
        CALL CreaFile
      CASE "E"
        CALL EsaminaFile
      CASE "S"
        CALL StampaFile
      CASE "A"
        CALL AggiungiFile
      CASE "U"
        END
      CASE ELSE
        LOCATE 21, 5: PRINT "Non e' stata premuto C, E, S, A od U."
        LOCATE 22, 5: PRINT "Premere un tasto per ritentare."
        ripetiz$ = INPUT$(1)
    END SELECT
    CLS 2
  LOOP
  END

SUB CreaFile STATIC
' Crea il file Notes.Dat. Ogni record e' una stringa

  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Crea un nuovo file chiamato Notes.Dat."
  Text$(3) = "Porre il disco dati nel drive B e premere un tasto."
  Text$(4) = "Al prompt (), digitare un record e premere Invio;"
  Text$(5) = "per finire il file premere Invio senza scrivere dati."
```

```
  CLS
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Attende che si prema un tasto per incominciare
  LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
  untasto$ = INPUT(1)

  ' Definisce le righe da 1 a 20 come finestra per introdurre i record
  VIEW PRINT 1 TO 20
  CLS 2

  ' Apre il file NotePad.Dat sull'unita' B per l'output come file #1
  OPEN "B:NotePad.Dat" FOR OUTPUT AS #1

  ' Introduce i record da tastiera e scrive nel file
  DO
    LINE INPUT "> "; record$
    PRINT
    IF record$ = "" THEN EXIT DO
    WRITE #1, record$
  LOOP

  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB

SUB EsaminaFile STATIC
' Percorre il file Notes.Dat, un record alla volta

  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Esamina il file Notes un record alla volta."
  Text$(3) = "Porre il disco dati nel drive B e premere un tasto:"
  Text$(4) = "verra' visualizzato il primo record nella finestra."
  Text$(5) = "Premere la barra spaziatrice per i successivi od U per uscire."
  CLS
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Definisce le righe da 1 to 20 come finestra per esaminare i record
  VIEW PRINT 1 TO 20

  ' Attende che venga premuto un tasto per incominciare
  LOCATE 10, 28:PRINT "Premere un tasto per iniziare"
  untasto$ = INPUT$(1)
  CLS 2                          'Pulisce solamente la finestra
```

```
  ' Apre il file Notes.Dat sull'unita' B per l'input come file 1
  OPEN "B:Notes.Dat" FOR INPUT AS #1

  ' Legge un record quando viene premuto un tasto diverso da U
  DO UNTIL EOF(1)
    LINE INPUT #1, record$
    PRINT record$
    PRINT
    tasto$ = INPUT$(1)
    IF UCASE$(tasto$) = "U" THEN EXIT DO
  LOOP

  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB

SUB StampaFile STATIC
' Stampa il file Notes.Dat

  ' Pone le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Invia il file Notes alla stampante."
  Text$(3) = "Porre il disco dei dati nel drive B."
  Text$(4) = "Assicurarsi che la stampante sia pronta e premere un tasto."
  Text$(5) = "Per interrompere premere il tasto U."
  CLS

  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Definisce le linee da 1 a 20 come finestra
  ' dove visualizzare i record durante la stampa
  VIEW PRINT 1 TO 20

  ' Attende la pressione di un tasto per cominciare
  LOCATE 10, 28:PRINT "Premere un tasto per iniziare"
  untasto$ = INPUT$(1)
  CLS 2                          'Cancella solo la finestra

  ' Apre il file Notes.Dat in input come file numero 1
  OPEN "B:Notes.Dat" FOR INPUT AS #1

  ' Legge i record, li visualizza e li stampa

  DO UNTIL EOF(1)
    tas$ = INKEY$: IF UCASE$(tas$) = "U" THEN EXIT DO
    INPUT #1, record$
    PRINT record$
```

```
    LPRINT record$
    LPRINT
  LOOP

  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB

SUB AggiungiFile STATIC
' Aggiunge informazioni al file Notes.Dat

  ' Pone le istruzioni nelle linee da 21 a 25

  Text$(1) = STRING$(74, 196)
  Text$(2) = "Aggiunge record al file Notes."
  Text$(3) = "Porre il disco dei dati nel drive B e premere un tasto."
  Text$(4) = "Al prompt (>), digitare un record e premere Invio."
  Text$(5) = "Premere Invio senza digitare alcun dato per terminare."
  CLS

  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Definisce le righe da 1 to 20 come finestra per i record da aggiungere
  VIEW PRINT 1 TO 20

  ' Attende la pressione di un tasto per iniziare
  LOCATE 10, 28:PRINT "Premere un tasto per iniziare."
  untasto$ = INPUT$(1)
  CLS 2                          'Cancella solo la finestra

  ' Apre il file Notes.Dat nel drive B per aggiungere come file numero 1
  OPEN "B:Notes.Dat" FOR APPEND AS #1

  ' Legge i record da tastiera e li aggiunge al file
  DO UNTIL EOF(1)
    LINE INPUT "#1"; record$
    PRINT
    IF record$ = "" THEN EXIT DO
    WRITE #1, record$
  LOOP

  ' Chiude il file e termina il sottoprogramma
  CLOSE #1
END SUB
```

Programma 6.5 NOTES CON MENU

6.8 Sommario

In questo capitolo si è presentata una trattazione generale sui file di dati seguita da informazioni specifiche riguardanti i file sequenziali; gran parte del capitolo è stata volta ai file sequenziali non strutturati.

Sono state date informazioni dettagliate sullo sviluppo di sottoprogrammi per eseguire quattro funzioni su un file sequenziale di dati non strutturato chiamato Notes:

- creare il file;
- leggere i record nel file;
- stampare i record del file;
- aggiungere appunti al file.

Il capitolo si è concluso con l'uso di tecniche di editing per includere in un programma integrato quattro programmi contenenti i sottoprogrammi. Il programma integrato esegue una qualunque delle quattro funzioni che possono essere selezionate da un menu.

Capitolo 7

File sequenziali strutturati

7.1 Stati ed abbrevazioni
7.2 Ricerca di informazioni nel file di esempio
7.3 Memorizzazione delle infomazioni
7.4 Sommario

Il capitolo precedente ha presentato l'uso dei file sequenziali non strutturati, nei quali ciascun record è una stringa a sé stante; il presente tratta invece quelli strutturati, nei quali ciascun record consiste di due o più campi. Tra i vari comandi e funzioni saranno utilizzati:

- OPEN e CLOSE per l'apertura e la chiusura dei file;
- PRINT # e WRITE # per memorizzare le informazioni nei file;
- INPUT # e LINE INPUT # per leggere le informazioni dai file;
- LOF per conoscere la lunghezza di un file.

Il capitolo prende spunto da un unico file strutturato i cui record consistono di due campi, entrambi contenenti stringhe; viene illustrato un programma di creazione e lettura dei contenuti e successivamente, tramite tre variazioni, vengono mostrati diversi metodi di lettura: tramite una facile sperimentazione si apprenderà come le istruzioni di output inviino le informazioni ai file dati e come quelle di input le ricevano da essi.

7.1 Stati ed abbreviazioni

Un buon programma applicativo che utilizza i file sequenziali è il programma presentato nel seguito che fornisce il codice postale (un'abbreviazione di due lettere) di qualsiasi stato

o territorio degli Stati Uniti; il file utilizzato è breve ed è difficile che le sue informazioni cambino nel tempo. Il Programma 7.1, STATI ED ABBREVIAZIONI, crea ed esamina un file sequenziale contenente i nomi degli stati e dei territori insieme ai loro codici postali; ciascun suo record comprende due campi.

Stato$ Un campo di lunghezza variabile da 4 a 32 caratteri contenente il nome dello stato o del territorio (ad esempio Iowa o California)

Abbr$ Un campo di lunghezza fissa di due caratteri contenente le lettere dell'abbreviazione postale (ad esempio IA)

Le informazioni da porre nel file sono contenute nelle istruzioni DATA del programma principale, di cui viene mostrata una parte:

```
DATA Alabama,AL,  Alaska,AK,  American Samoa,AS
DATA Arizona,AZ,  Arkansas,AR,  California,CA+
DATA Colorado,CO,  Connecticut,CT,  Delaware,DE

  .

  .

  .

DATA Wisconsis,WI,  Wyoming,WY
DATA Fine dei dati,ZZ
```

Digitando questo listato è possibile utilizzare solo una parte delle istruzioni DATA presentate nel testo, ad esempio tre o quattro; in ogni caso ci si assicuri di includere l'ultima istruzione DATA:

```
DATA Fine dei dati,ZZ


DECLARE SUB EsaminaFile ()
REM ** STATI ED ABBREVIAZIONI - CREAZIONE E PROVA DEL FILE **
' File: UQBO701.BAS

REM ** Crea il file **
' Crea il file StatiAbr.Dat con i nomi degli stati e le abbreviazioni
' File sequenziale con due campi: Stato$, Abbr$
CLS : DEFINT A-Z

OPEN "B:StatiAbr.Dat" FOR OUTPUT AS #1
DO
   READ Stato$, Abbr$            'Legge i dati di un record
   IF Abbr$ = "ZZ" THEN EXIT DO  'Esce al termine della lettura dei record
   WRITE #1, Stato$, Abbr$       'Scrive un record nel file
LOOP
CLOSE #1

DATA Alabama,AL, Alaska,AK, American Samoa,AS
DATA Arizona,AZ, Arkansas,AR, California,CA
```

```
DATA Colorado,CO, Connecticut,CT, Delaware,DE
DATA District of Columbia,DC, Federate States Of Micronesia,TT
DATA Florida,FL, Georgia,GA, Guam,GU
DATA Hawaii,HI, Idaho,ID, Illinois,IL
DATA Indiana,IN, Iowa,IA, Kansas,KS
DATA Kentucky,KY, Louisiana,LA, Maine,ME
DATA Mariana Islands,CM, Marshall Islands,TT, Maryland,MD
DATA Massachusetts,MA, Michigan,MI, Micronesia,TT
DATA Minnesota,MN, Mississippi,MS, Missouri,MO
DATA Montana,MT, Nebraska,NE, Nevada,NV
DATA New Hampshire,NH, New Jersey,NJ, New Mexico,NM
DATA New York,NY, North Carolina,NC, North Dakota,ND
DATA Northern Mariana Island,CM, Ohio,OH, Oklahoma,OK
DATA Oregon,OR, Palau,TT, Pennsylvania,PA
DATA Puerto Rico,PR, Rhode Island,RI, South Carolina,SC
DATA South Dakota,SD, Tennessee,TN, Texas,TX
DATA Utah,UT, Vermont,VT, Virgin Islands,VI
DATA Virginia,VA, Washington,WA, West Virginia,WV
DATA Wisconsin,WI, Wyoming,WY
DATA Fine dei dati,ZZ

REM ** Esamina il file e lo verifica, quindi termina **
CALL EsaminaFile
END

SUB EsaminaFile STATIC
' Esamina il file StatiAbr.Dat, un record alla volta
  ' Scrive le istruzioni nelle righe da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Esamina il file degli Stati e delle Abbreviazioni."
  Text$(3) = "Porre il disco dei dati nel drive B."
  Text$(4) = "Premere un tasto per iniziare: apparira' un record."
  Text$(5) = "Premere la barra spaziatrice per avanzare, U per uscire."
  CLS
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Definisce le righe da 1 a 20 come finestra di visualizzazione dei record
   VIEW PRINT 1 TO 20
   ' Attende la pressione di un tasto per iniziare
   PRINT "Premere un tasto per iniziare."
   untasto$ = INPUT$(1)
   CLS 2                          'Cancella la finestra di visualizzazione

   ' Apre in input il file StatiAbr.Dat del drive B con il numero 1
   OPEN "B:StatiAbr.Dat" FOR INPUT AS #1

   ' Legge un record ad ogni pressione di tasto diverso da U
```

```
  DO UNTIL EOF(1)
    LINE INPUT #1, record$
    PRINT record$
    PRINT
    prossimo$ = INPUT$(1)
    IF UCASE$(prossimo$) = "U" THEN EXIT DO
  LOOP

  ' Chiude il file e termina il sottoprogramma
  VIEW PRINT 1 TO 25: CLS

  CLOSE #1
END SUB
```

Programma 7.1 STATI ED ABBREVIAZIONI - FILE DI CREAZIONE E PROVA

Il Programma 7.1 crea un file leggendo i record dalle istruzioni DATA e scrivendoli in esso all'interno di una struttura DO. . .LOOP che termina alla lettura dello stato fittizio Fine dei dati, ZZ, come mostrato nel seguente frammento di programma:

```
DO
  READ Stato$, Abbr$              'Legge i dati di un record
  IF Abbr$ = "ZZ" THEN EXIT DO  'Esce al termine della lettura dei record
  WRITE #1, Stato$, Abbr$         'Scrive un record nel file
LOOP
```

Il nome dello stato o territorio viene assegnato alla variabile Stato$, mentre l'abbreviazione è assegnata ad Abbr$; i valori delle due variabili sono letti dall'istruzione DATA e scritti nel file in un record composto da due campi, chiamati anch'essi Stato$ ed Abbr$.

Si noti che l'istruzione WRITE # sotto mostrata separa i campi scritti nel file con virgole e racchiude le stringhe tra doppie virgolette; inoltre esso pone un ritorno carrello ed un avanzamento riga dopo l'ultimo campo di un record.

```
            WRITE #1, Stato$, Abbr$
invia un record   ^      ^  -  ^ il record è formato
al file aperto come #1            da questi due campi
```

Per questo motivo un record del file creato dal Programma 7.1 comprende:

- il valore di Stato$ racchiuso tra virgolette;
- una virgola;
- il valore di Abbr$ racchiuso tra virgolette;
- i caratteri ritorno carrello ed avanzamento riga che indicano la fine del record.

Eseguendo il Programma 7.1 ed esaminando il file si potranno vedere le virgole e le stringhe racchiuse tra virgolette; dopo avere creato i record ed averli scritti sul file, il programma chiama il sottoprogramma EsaminaFile, la cui schermata iniziale appare come mostrato nella Figura 7.1.

```
                    Premere un tasto per iniziare


Esamina il file degli Stati e delle Abbreviazioni.
Porre il disco dei dati nel drive B.
Premere un tasto per iniziare: apparira' un record.
Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 7.1 Parte iniziale dell'output di EsaminaFile

Si prema un tasto qualsiasi per iniziare a verificare i contenuti del file: la Figura 7.2 mostra una sessione di verifica del file in corso con i primi nove record visualizzati sullo schermo.

```
"Alabama","AL"
"Alaska","AK"
"American Samoa","AS"
"Arizona","AZ"
"Arkansas","AR"
"California","CA"
"Colorado","CO"
"Connecticut","CT"
"Delaware","DE"

Esamina il file degli Stati e delle Abbreviazioni.
Porre il disco dei dati nel drive B.
Premere un tasto per iniziare: apparira' un record.
Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 7.2 Sessione di verifica dei contenuti del file

Il sottoprogramma EsaminaFile legge un record completo comprendente entrambi i campi (Stato$ ed Abbr$) servendosi di un'istruzione LINE INPUT # e li scrive servendosi di una sola stringa (record$).

```
LINE INPUT #1, record$
PRINT record$
```

L'istruzione LINE INPUT legge tutti i caratteri, incluse le virgolette, fino ad incontrare il ritorno carrello e l'avanzamento riga che indicano la fine di un record, per cui, durante

l'esame dei contenuti di un file appariranno le virgolette e la virgola inserite dall'istruzione WRITE # durante la creazione del file, visualizzando la codifica del record all'interno del file.

Continuando a premere la **BARRA SPAZIATRICE** verranno mostrati sullo schermo nuovi stati o territori con le loro abbreviazioni; una volta che non rimanga più spazio sullo schermo, la successiva pressione della **BARRA SPAZIATRICE** farà scorrere il testo visualizzato nell'area superiore verso l'alto per fare spazio ai nuovi dati, lasciando ferme le istruzioni fornite dal sottoprogramma. Quando compare "Wyoming", come mostrato nella Figura 7.3, è stata raggiunta la fine del file.

Premendo la **BARRA SPAZIATRICE** od il tasto **U**, il programma termina: lo schermo viene cancellato e compare il messaggio "Premere un tasto per continuare"; premendo un tasto si tornerà alla schermata di Modifica di QuickBASIC.

```
"Texas","TX"
"Utah","UT"
"Vermont","VT"
"Virgin Islands","VI"
"Virginia","VA"
"Washington","WA"
"West Virginia","WV"
"Wisconsin","WI"
"Wyoming","WY"

Esamina il file degli Stati e delle Abbreviazioni.
Porre il disco dei dati nel drive B.
Premere un tasto per iniziare: apparira' un record.
Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 7.3 Parte finale dell'output di EsaminaFile

A questo punto si modifichino le due righe del DO. . .LOOP del sottoprogramma EsaminaFile in modo che un record sia letto come due campi da un'istruzione INPUT # e sia scritto sullo schermo come due stringhe separate:

```
DO. . .LOOP originale

  ' Legge un record ad ogni pressione di tasto diverso da U
  DO UNTIL EOF(1)
    LINE INPUT #1, record$
    PRINT record$
```

```
    PRINT
    prossimo$ = INPUT$(1)
    IF UCASE$(prossimo$) = "U" THEN EXIT DO
  LOOP

DO. . .LOOP modificato
  ' Legge un record ad ogni pressione di tasto diverso da U
  DO UNTIL EOF(1)
    INPUT #1, Stato$, Abbr$
    PRINT Stato$, Abbr$
    PRINT
    prossimo$ = INPUT$(1)
    IF UCASE$(prossimo$) = "U" THEN EXIT DO
  LOOP
```

L'istruzione INPUT # leggerà il primo campo del record ed assegnerà il suo valore a Stato$, quindi il secondo ed assegnerà il suo valore ad Abbr$; i due campi sono separati nel record da una virgola: l'istruzione INPUT # riconosce questa virgola come separatore dei campi e non include tale virgola, né le virgolette, durante l'assegnamento dei valori alle variabili.

La Figura 7.4 mostra i primi nove stati e le loro abbreviazioni visualizzati dall'esecuzione del programma modificato: confrontandoli con l'output del Programma 7.1 nella Figura 7.2 si noterà che l'abbreviazione viene mostrata in una zona dello schermo differente da quella del nome dello stato, dal momento che l'istruzione PRINT # per la scrittura su schermo utilizza una virgola come spaziatore.

```
Alabama        AL

Alaska         AK

American Samoa                    AS

Arizona        AZ

Arkansas       AR

California     CA

Colorado       CO

Connecticut    CT

Delaware       DE


Esamina il file degli Stati e delle Abbreviazioni.

Porre il disco dei dati nel drive B.

Premere un tasto per iniziare: apparira' un record.

Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 7.4 Parte iniziale delll'output di EsaminaFile modificato

Si modifichi l'istruzione PRINT come mostrato nella seguente struttura DO. . .LOOP:

```
' Legge un record ad ogni pressione di tasto diverso da U.
DO UNTIL EOF(1)
  INPUT #1, Stato$, Abbr$
  PRINT Stato$; Abbr$
  PRINT
  prossimo$ = INPUT$(1)
  IF UCASE$(prossimo$) = "U" THEN EXIT DO
LOOP
```

Eseguendo il programma così modificato, i nomi dei primi nove record appariranno sullo schermo come illustrato nella Figura 7. 5; si noti che non vi sono spazi tra gli stati e le loro abbreviazioni.

```
AlabamaAL

AlaskaAK

American SamoaAS

ArizonaAZ

ArkansasAR

CaliforniaCA

ColoradoCO

ConnecticutCT

DelawareDE


Esamina il file degli Stati e delle Abbreviazioni.

Porre il disco dei dati nel drive B.

Premere un tasto per iniziare: apparira' un record.

Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 7.5 Output di EsaminaFile con un punto e virgola nell'istruzione PRINT

Un'altra variazione: si utilizzi TAB per scrivere stati ed abbreviazioni in colonne separate, cambiando il DO. . .LOOP in:

```
' Legge un record ad ogni pressione di tasto diverso da U
DO UNTIL EOF(1)
  INPUT #1, Stato$, Abbr$
  PRINT Stato$; TAB(35); Abbr$
  PRINT
  prossimo$ = INPUT$(1)
  IF UCASE$(prossimo$) = "U" THEN EXIT DO
LOOP
```

La funzione TAB(35) fa si che la scrittura delle abbreviazioni inizi alla colonna 35 (si ricordi che il nome più lungo è di 34 caratteri): i nomi e le loro abbreviazioni sono scritti in colonne separate, come mostrato nella Figura 7.6

```
Alabama                    AL
Alaska                     AK
American Samoa             AS
Arizona                    AZ
Arkansas                   AR
California                 CA
Colorado                   CO
Connecticut                CT
Delaware                   DE

Esamina il file degli Stati e delle Abbreviazioni.
Porre il disco dei dati nel drive B.
Premere un tasto per iniziare: apparira' un record.
Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 7.6 Output di EsaminaFile con TAB nell'istruzione PRINT

Sarebbe opportuno salvare questa come versione definitiva del Programma 7.1; se l'elenco dovesse subireai variazioni, basterebbe modificare il programma e creare un nuovo file.

Si può compiere un'altra serie di esperimenti come segue.

- Si cambi l'istruzione WRITE # del programma principale con PRINT # nel Programma 7-1:

  ```
  Si cambi     WRITE #1, Stato$, Abbr$
  in           PRINT #1, Stato$, Abbr$
  ```

  Si provi ad applicare questa variazione ad alcuni od a tutti i metodi di lettura del file mostrati in precedenza.

- Si utilizzino dei delimitatori espliciti nell'istruzione PRINT #; si aggiunga una riga prima del DO. . .LOOP e lo si modifichi come segue:

  ```
  Q$ = CHR$(34)
  DO
   READ Stato$, Abbr$
   IF Abbr$ = "ZZ" THEN EXIT DO
   PRINT #1, Q$ Stato$ Q$, Q$ Abbr$ Q$
  LOOP
  ```

Una volta spostato il cursore dalla linea contenente l'istruzione PRINT # si vedrà che QuickBASIC trasforma la linea in

```
PRINT #1, Q$; Stato$; Q$, Q$; Abbr$; Q$
```

Non è necessario allarmarsi: QuickBASIC alcune volte modifica il formato degli elementi del programma. Si provi questa tecnica con alcune delle variazioni del sottoprogramma EsaminaFile.

- Si possono compiere altri esperimenti togliendo la virgola dall'istruzione PRINT #:

```
PRINT #1, Q$ Stato$ Q$ Q$ Abbr$ Q$
```

QuickBASIC modificherà il formato dell'istruzione in:

```
PRINT #1, Q$; Stato$; Q$; Q$; Abbr$; Q$
```

Anche questo metodo può essere verificato in alcune od in tutte le variazioni di EsaminaFile.

Sono stati qui trattati alcuni metodi per salvare informazioni nel file degli stati e delle abbreviazioni e per leggere tali informazioni dal file stesso; per la convenienza del lettore di seguito sono riepilogati tutti i metodi esaminati.

Salvataggio delle informazioni nel file

```
WRITE #1, Stato$, Abbr$
PRINT #1, Stato$, Abbr$
WRITE #1, Q$; Stato$; Q$, Q$; Abbr$; Q$
PRINT #1, Q$; Stato$; Q$, Q$; Abbr$; Q$
```

dove:

```
Q$ = CHR$(34)
```

Lettura delle informazioni dal file

```
LINE INPUT #1, record$
INPUT #1, Stato$, Abbr$
```

Si provino le varie combinazioni di tali metodi.

7.2 Ricerca di informazioni nel file di esempio

Il Programma 7.2, RICERCA DELLE ABBREVIAZIONI POSTALI, esamina il file StatiAbr.Dat alla ricerca di un record contenente un particolare stato o territorio servendosi di una chiave di ricerca inserita dalla tastiera.

Questa ricerca viene compiuta dalle linee del programma principale riportate nella pagina seguente.

```
'Legge la chiave di ricerca. La stringa vuota fa terminare
DO
  INPUT "Stato o Territorio"; Ricerca$
  IF Ricerca$ = "" THEN
    VIEW PRINT 1 TO 25: CLS
    EXIT DO
  END IF
```

Si può far terminare l'esecuzione del programma immettendo una stringa vuota invece della chiave di ricerca; si prema semplicemente **INVIO** senza scrivere nulla: lo schermo verrà ripulito, eccezione fatta per la scritta "Premere un tasto per continuare" in basso: premendo un tasto QuickBASIC ritornerà alla schermata di Modifica. Nel caso sia digiti una chiave di ricerca, il file StatiAbr.Dat sarà aperto e verrà chiamato il sottoprogramma RicercaFile.

```
'Cerca e visualizza i record che contengono Ricerca$
OPEN "B:StatiAbr.Dat" FOR INPUT AS #1
CALL RicercaFile(Ricerca$)
 DECLARE SUB RicercaFile (Chiave$)
 REM ** RICERCA DELLE ABBREVIAZIONI POSTALI **
 ' File: UQB0702.BAS
 ' Trova l'abbreviazione postale di qualsiasi stato o territorio nel
 ' file StatiAbr.Dat; la chiave di ricerca puo' essere la parte iniziale
 ' del nome dello stato o territorio; vengono trovati tutti i record che
 ' iniziano con la chiave data, che puo' essere maiuscola, minuscola o
 ' combinazione di lettere maiuscole e minuscole
 ' Scrive le istruzioni nelle righe da 21 a 25
 Text$(1) = STRING$(74, 196)
 Text$(2) = "Porre il disco con il file StatiAbr.Dat nel drive B"
 Text$(3) = "e premere un tasto per iniziare."
 Text$(4) = "Al prompt, digitare la chiave di ricerca e premere INVIO."
 Text$(5) = "Per uscire, premere INVIO senza avere immesso alcun dato."
 CLS : DEFINT A-Z
 FOR riga = 1 TO 5
   LOCATE riga + 20, 3: PRINT Text$(riga);
 NEXT riga
 ' Definisce le righe da 1 a 20 come finestra di visualizzazione
 VIEW PRINT 1 TO 20
 ' Attende la pressione di un tasto
 LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
 untasto$ = INPUT$(1)
 CLS 2
 ' Legge la chiave di ricerca; se vuota, termina il programma
 DO
   INPUT "Stato o territorio"; Ricerca$
   IF Ricerca$ = "" THEN
     VIEW PRINT 1 TO 25: CLS
     EXIT DO
```

```
  END IF
  ' Cerca e visualizza i record che contengono Ricerca$
  OPEN "B:StatiAbr.Dat" FOR INPUT AS #1
  CALL RicercaFile(Ricerca$)
  CLOSE #1
  PRINT
LOOP
END

SUB RicercaFile (Chiave$)
' Compie la ricerca nel file StatiAbr.Dat di tutti i record la cui
' parte iniziale di Stato$ sia uguale a Chiave$; in caso non ne siano
' trovati, emette il messaggio: Stato non trovato.
  Chiave$ = UCASE$(Chiave$)
  LunChiave = LEN(Chiave$)
  trovato = 0                          'Per ora nessuno corrisponde
  DO UNTIL EOF(1)
    INPUT #1, Stato$, Abbr$            'Legge un record
    Prova$ = UCASE$(LEFT$(Stato$, LunChiave))
    IF Prova$ = Chiave$ THEN
      trovato = -1                     'Trovato
      PRINT Stato$; TAB(35); Abbr$
    ELSEIF Prova$ > Chiave$ THEN
      EXIT DO
    END IF
  LOOP
  IF trovato = 0 THEN PRINT "Stato non trovato."
END SUB
```

Programma 7.2 RICERCA DELLE ABBREVIAZIONI POSTALI

La variabile Ricerca$, contenente la chiave di ricerca, viene passata al sottoprogramma alla sua chiamata, diviene il valore di Chiave$ e le sue lettere sono rese maiuscole.

```
Chiave$ = UCASE$(Chiave$)
```

Quindi RicercaFile calcola la lunghezza di Chiave$ ed azzera il flag trovato per indicare che non sono state ancora trovate corrispondenze.

```
LunChiave = LEN(Chiave$)
trovato = 0
```

Un DO. . .LOOP legge i record del file StatiAbr.Dat confrontando la chiave di ricerca con la variabile Prova$ che contiene i primi n caratteri di ciascuno stato, dove n è il numero pari alla lunghezza della chiave di ricerca fornita; se lo stato letto dal record è maggiore di quello richiesto in ordine alfabetico, un'istruzione EXIT DO termina la ricerca uscendo dal ciclo.

```
DO UNTIL EOF(1)
  INPUT #1, Stato$, Abbr$                'Legge un record
  Prova$ = UCASE$(LEFT$(Stato$, LunChiave))
  IF Prova$ = Chiave$ THEN
    trovato = -1                         'Trovato
    PRINT Stato$; TAB(35); Abbr$
  ELSEIF Prova$ > Chiave$ THEN
    EXIT DO
  END IF

LOOP
IF trovato = 0 THEN PRINT "Stato non trovato."
```

Il DO. . .LOOP scrive sullo schermo ciascun record corrispondente alla chiave di ricerca; se non ne viene trovato nessuno, il valore di trovato rimane zero, nel qual caso viene scritto il messaggio "Stato non trovato" al momento dell'uscita dal DO. . .LOOP.Al termine del sottoprogramma il controllo viene restituito al programma principale, dove il file viene chiuso e viene ripetuto il ciclo per consentire l'immissione di una nuova chiave di ricerca o la terminazione del programma.

```
' Legge la chiave di ricerca; se vuota, termina il programma
DO
  INPUT "Stato o territorio"; Ricerca$
  IF Ricerca$ = "" THEN
    VIEW PRINT 1 TO 25: CLS
    EXIT DO
  END IF

  ' Ricerca e visualizza i record contenenti Ricerca$
  OPEN "B:StatiAbr.Dat" FOR INPUT AS #1
  CALL RicercaFile(Ricerca$)
  CLOSE #1
  PRINT
LOOP
END
```

Per utilizzare il Programma 7.2 si digiti il nome dello stato o del territorio desiderato: verrà trovato lo stato e la sua abbreviazione; se si iserisce solo la prima lettera, verrannot riportati tutti gli stati ed i territori che iniziano con tale lettera. Si può anche provare a digitare le prime due o tre lettere del nome dello stato: il programma troverà tutti i record le cui prime lettere del nome dello stato o del territorio corrispondono a quelle fornite.

Eseguendo il Programma 7.2 viene visualizzata la schermata d'apertura (Figura 7.7) e premendo un tasto viene richiesto uno stato od un territorio nella parte alta dello schermo:

```
Stato o Territorio? _
```

Le istruzioni d'uso del programma appaiono nella parte bassa dello schermo.

```
Porre il disco con il file StatiAbr.Dat nel drive B
e premere un tasto qualsiasi per iniziare.
Digitare al prompt la chiave di ricerca e premere INVIO.
Per terminare premere INVIO senza scrivere alcun dato.
```

Il nome da ricercare può essere digitato combinando in qualsiasi modo lettere maiuscole e minuscole; nella Tabella 7-1 sono mostrati alcuni esempi di input e dei risultati corrispondenti, mentre la Figura 7.8 mostra alcuni esempi di esecuzione del programma.

```
Premere un tasto per iniziare
```

```
Porre il file degli stati ed abbreviazioni nel drive B
e premere un tasto per iniziare.
Al prompt, digitare la chiave di ricerca e premere INVIO.
Per uscire, premere INVIO senza immettere alcun dato.
```

Figura 7.7 Schermata d'apertura del Programma 7.2

Tabella 7.1 Esempi d'utilizzo del Programma 7.2

Input	Risultato
C	California CA
	Colorado CO
	Connecticut CT
nor	North Carolina NC
	North Dakota ND
	Northern Mariana Islands CM
VIRGIN	Virgin Islands VI
	Virginia VA
Ioway	Stato non trovato
Iowa	Iowa IA

```
Stato o Territorio? a
Alabama                    AL
Alaska                     AK
American Samoa             AS
```

```
Arizona                          AZ
Arkansas                         AR

Stato o territorio? z
Stato non trovato

Stato territorio? nor
North Carolina                   NC
North Dakota                     ND
Northern Mariana Islands         CM

Stato o Territorio? _

Porre il file degli stati ed abbreviazioni nel drive B
e premere un tasto per iniziare.
Al prompt, digitare la chiave di ricerca e premere INVIO.
Per uscire, premere INVIO senza immettere alcun dato.
```

Figura 7.8 Output del Programma 7.2

7.3 Memorizzazione delle informazioni

Per apprendere maggiori informazioni sul metodo di codifica delle informazioni memorizzate nei file si può utilizzare il Programma 7.3, ESPERIMENTI DI I/O SU FILE, che utilizza due file, Prova.Uno e Prova.Due, compiendo le seguenti operazioni:

- vengono aperti in output entrambi i file;
- viene letta una stringa digitata da tastiera in un record da un'istruzione LINE INPUT
- viene memorizzato il record in Prova.Uno per mezzo di un'istruzione PRINT #;
- viene scritto lo stesso record in Prova.Due utilizzando un'istruzione WRITE #;
- viene visualizzato sullo schermo il numero di caratteri di ciascun file servendosi di un'istruzione PRINT che utilizza la function LOF;
- vengono chiusi i due i file dall'istruzione CLOSE #1, #2;

- vengono aperti in input entrambi i file;
- viene letto e visualizzato sullo schermo il record di Prova.Uno per mezzo di un'istruzione LINE INPUT # .
- viene analogamente letto e visualizzato il record di Prova.Due da parte di un'istruzione LINE INPUT #;
- vengono chiusi entrambi i file con CLOSE #1, #2;
- dopo avere premuto un tasto qualsiasi, l'utente può digitare un'altra stringa.

```
REM ** ESPERIMENTI DI I/O SU FILE **
' File: UQB0703.BAS

' Esperimenti di input e output su file,
' utilizzando i file Prova.Uno e Prova.Due
DO
  OPEN "B:Prova.Uno" FOR OUTPUT AS #1
  OPEN "B:Prova.Due" FOR OUTPUT AS #2

  ' Legge i dati dalla tastiera e li memorizza
  ' nei due file in due modi differenti
  CLS
  LINE INPUT "Stringa, prego"; stringa$
  PRINT #1, stringa$
  WRITE #2, stringa$

  ' Visualizza la lunghezza di ciascuno dei file
  ' e li chiude.
  PRINT
  PRINT "Prova.Uno ha"; LOF(1); "caratteri."
  PRINT "Prova.Due ha"; LOF(2); "caratteri."
  CLOSE #1, #2
  ' Legge le informazioni di entrambi i file e li visualizza
  OPEN "B:Prova.Uno" FOR INPUT AS #1
  OPEN "B:Prova.Due" FOR INPUT AS #2
  PRINT : PRINT "Contenuti di Prova.Uno utilizzando LINE INPUT # :"
  LINE INPUT #1, record$
  PRINT record$
  PRINT : PRINT "Contenuti di Prova.Due utilizzando LINE INPUT # :"
  LINE INPUT #2, record$
  PRINT record$
  CLOSE #1, #2
  ' Indica come immettere un nuovo record
  PRINT : PRINT "Premere un tasto per continuare."
  untasto$ = INPUT$(1)
LOOP
END
```

Programma 7.3 ESPERIMENTI DI I/O SU FILE

Negli esempi che seguono si noti il numero di caratteri di ciascun file e le somiglianze e differenze di codifica del record nei due file. Si esegua il Programma 7.3 digitando una stringa di una sola lettera in risposta al prompt, ad esempio:

```
Stringa, prego? a

Prova.Uno ha 3 caratteri.
Prova.Due ha 5 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
a

Contenuti di Prova.Due utilizzando LINE INPUT # :
"a"

Premere un tasto per continuare
```

Esaminando i risultati, in Prova.Uno l'istruzione PRINT # ha memorizzato la stringa (a) nel file senza racchiuderla tra virgolette, per cui il file contiene tre caratteri, la lettera a, nonché il ritorno carrello e l'avanzamento riga che indicano la fine del record.

In Prova.Due l'istruzione WRITE # ha scritto invece parte della stringa (a) nel file racchiudendola tra virgolette ed il file contiene cinque caratteri: virgolette, lettera a, virgolette, ritorno carrello ed avanzamento riga. Si provi un altro esempio:

```
Stringa, prego? 123

Prova.Uno ha 5 caratteri.
Prova.Due ha 7 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
123

Contenuti di Prova.Due utilizzando LINE INPUT # :
"123"
Premere un tasto per continuare
```

In questo caso sono stati digitati tre caratteri (123): Prova.Uno contiene cinque caratteri: 1, 2, 3, ritorno carrello ed avanzamento riga; Prova.Due contiene sette caratteri: virgolette, 1, 2, 3, virgolette, ritorno carrello ed avanzamento riga. Si provi infine un'ultimo esempio:

```
Stringa, prego? a  ,  b  ,  c  ,  1  ,  2  ,  3

Prova.Uno ha 43 caratteri.
Prova.Due ha 45 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
a  ,  b  ,  c  ,  1  ,  2  ,  3

Contenuti di Prova.Due utilizzando LINE INPUT # :
"a  ,  b  ,  c  ,  1  ,  2  ,  3"

Premere un tasto per continuare
```

Anche in questo caso entrambi i file contengono la stessa informazione codificata alla stesso modo, eccettuate le virgolette introdotte dall'istruzione WRITE # nella scrittura su Prova.Due.

7.3.1 Modifica del programma

Si modifichi il programma in modo che il record da scrivere nei due file sia immesso in due stringhe separate e scritto nei file in un record a due campi, modificando la seguente porzione del listato:

```
' Legge i dati dalla tastiera e li memorizza
' nei due file in due modi differenti
CLS
LINE INPUT "Stringa, prego"; stringa$
PRINT #1, stringa$
WRITE #2, stringa$
```

in:

```
' Legge i dati dalla tastiera e li memorizza
' nei due file in due modi differenti
CLS
LINE INPUT "Prima e seconda stringa, prego"; PrimaStringa$, SecondaStringa$
```

```
PRINT #1, PrimaStringa$, SecondaStringa$
WRITE #2, PrimaStringa$, SecondaStringa$
```

Dopo avere effettuato i cambiamenti, il Programma 7.3 scriverà un record composto da due campi, che verrà poi letto nella sua esatta codifica dall'istruzione LINE INPUT #.

LINE INPUT # legge i record interi, con tutti i loro campi, fino ai caratteri ritorno carrello ed all'avanzamento riga che ne indicano la fine.

Si esegua il programma modificato, provando il seguente esempio:

```
Prima e seconda stringa, prego? a,b

Prova.Uno ha 17 caratteri.
Prova.Due ha 9 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
a             b

Contenuti di Prova.Due utilizzando LINE INPUT # :
"a","b"

Premere un tasto per continuare
```

Prova.Uno è adesso lungo otto caratteri più di Prova.Due; i caratteri extra sono spazi ed inoltre durante la scrittura del record in Prova.Due l'istruzione WRITE # ha racchiuso le stringhe tra virgolette e le ha separate con una virgola. Ecco altri tre esempi:

```
Prima e seconda stringa, prego? "a,b",c

Prova.Uno ha 17 caratteri.
Prova.Due ha 11 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
a,b           c

Contenuti di Prova.Due utilizzando LINE INPUT # :
"a,b","c"
```

```
Premere un tasto per continuare

Prima e seconda stringa, prego? 1234567890123,abc

Prova.Uno ha 19 caratteri.
Prova.Due ha 23 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
1234567890123 abc

Contenuti di Prova.Due utilizzando LINE INPUT # :
"1234567890123","abc"

Premere un tasto per continuare

Prima e seconda stringa, prego? 12345678901234,abc

Prova.Uno ha 33 caratteri.
Prova.Due ha 24 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
12345678901234                    abc

Contenuti di Prova.Due utilizzando LINE INPUT # :
"12345678901234","abc"

Premere un tasto per continuare
```

Gli ultimi due esempi dimostrano come l'istruzione PRINT #, se si utilizza una virgola per separare i nomi delle variabili, scrive i dati nel file nello stesso modo utilizzato per visualizzarli sullo schermo.

Ciò può essere più facilmente compreso modificando l'istruzione PRINT # del programma in:

```
PRINT #1, PrimaStringa$; SecondaStringa$
                         ^ si usi un punto e virgola
```

Un esempio di esecuzione dopo questa modifica produce:

```
Prima e seconda stringa, prego? a,b

Prova.Uno ha 4 caratteri.
Prova.Due ha 9 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
ab
Contenuti di Prova.Due utilizzando LINE INPUT # :
"a","b"

Premere un tasto per continuare
```

La prossima variazione del Programma 7.3 mostra come sono memorizzati i numeri nei file; si modifichino come indicatori di seguito le parti del programma che leggono i dati dalla tastiera e li scrivono nei due file:

```
' Legge i dati dalla tastiera e li memorizza
' nei due file in due modi differenti
CLS
INPUT "Primo e secondo numero, prego", PrimoNumero, SecondoNumero
PRINT #1, PrimoNumero, SecondoNumero
WRITE #1, PrimoNumero, SecondoNumero
```

Si provi quindi questo esempio:

```
Primo e secondo numero, prego? 1,2

Prova.Uno ha 19 caratteri.
Prova.Due ha 5 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
1            2

Contenuti di Prova.Due utilizzando LINE INPUT # :
1,2
Premere un tasto per continuare
```

Durante la scrittura in Prova.Uno, l'istruzione PRINT # scrive:

- il primo numero nella forma spazio, cifra, spazio;
- 11 spazi per un totale di 14 caratteri compressivi, seguendo il formato standard del separatore virgola;
- il secondo numero nella forma spazio, cifra, spazio;
- i caratteri ritorno carrello ed avanzamento riga per indicare la fine del record.

Sommando tutti i caratteri sopra elencati si ottengono 19 caratteri in totale. L'istruzione WRITE # che scrive i numeri in Prova.Due non racchiude i numeri tra virgolette e si limita a separarli con una virgola; in entrambi i casi i numeri sono memorizzati come caratteri ASCII, uno per ogni cifra. Ecco altri tre esempi per completare la panoramica:

```
Primo e secondo numero, prego? 1   ,   2

Prova.Uno ha 19 caratteri.
Prova.Due ha 5 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
 1            2

Contenuti di Prova.Due utilizzando LINE INPUT # :
1,2

Premere un tasto per continuare

Primo e secondo numero, prego? 1234567,123

Prova.Uno ha 21 caratteri.
Prova.Due ha 13 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
 1234567      123

Contenuti di Prova.Due utilizzando LINE INPUT # :
1234567,123
Premere un tasto per continuare
```

```
Primo e secondo numero, prego? 12345678,123

Prova.Uno ha 35 caratteri.
Prova.Due ha 18 caratteri.

Contenuti di Prova.Uno utilizzando LINE INPUT # :
1.234567E+07                    123

Contenuti di Prova.Due utilizzando LINE INPUT # :
1.234567E+07,123

Premere un tasto per continuare
```

Seguono alcuni suggerimenti per eventuali esperimenti che vengono lasciati al lettore.Nell'ultima variazione, si provi a separare gli argomenti dell'istruzione PRINT # con il punto e virgola, come in:

```
PRINT #1, PrimoNumero; SecondoNumero
```

Invece di usare nel record campi contenenti due stringhe o due numeri, si provi ad avere un campo di ciascun tipo, ad esempio:

```
' Legge i dati dalla tastiera e li memorizza
' nei due file in due modi differenti.
CLS
INPUT "Stringa e numero, prego", Stringa$, Numero
PRINT #1, Stringa$, Numero
WRITE #1, Stringa$, Numero
```

7.4 Sommario

In questo capitolo è stato presentato l'utilizzo dei file sequenziali strutturati per mezzo delle istruzioni:

- OPEN #, per aprire i file;
- PRINT e WRITE #, per memorizzare le informazioni;
- INPUT # e LINE INPUT #, per recuperare le informazioni;
- LOF, per conoscere la lunghezza di un file;

- CLOSE #, per chiudere i file.

Si è appreso come creare ed esaminare file sequenziali strutturati composti da record con campi contenenti solo numeri, sole stringhe ed infine combinazioni dei due tipi.

Due programmi STATI ED ABBREVIAZIONI e RICERCA IL CODICE POSTALE sono stati utilizzati per dimostrare differenti approcci alla scrittura e lettura di dati in file sequenziali strutturati, mentre il terzo programma del capitolo, ESPERIMENTI DI I/O SU FILE ha consentito di sperimentare numerosi metodi di memorizzazione di dati su file.

Capitolo 8

File ad accesso casuale

8.1 Fondamenti
8.2 Creazione di un file ad accesso casuale
8.3 Lettura di un file ad accesso casuale
8.4 Esempio: valore nutrizionale degli alimenti
8.5 Sommario

Dopo avere appreso come padroneggiare l'uso dei file sequenziali negli ultimi due capitoli, in questo si affronterà la creazione, l'esame dei contenuti, la ricerca e la visualizzazione dei file ad accesso casuale, utilizzando le seguenti istruzioni e function:

OPEN #	Indica l'apertura di un file FOR RANDOM (ad accesso casuale) che consente la scrittura o la lettura dei dati dal file.
LEN =	Indica una clausola dell'istruzione OPEN che specifica la lunghezza dei record del file.
TYPE...END TYPE	Struttura che definisce i tipi delle variabili utilizzate nei file ad accesso casuale.
PUT #	Scrive un record in un file ad accesso casuale.
GET #	Legge un record da un file ad accesso casuale.
CLOSE	Chiude uno specifico file ad accesso casuale; se non viene specificato un numero di file vengono chiusi tutti i file aperti.
LOF	Restituisce la lunghezza di un file.

8.1 Fondamenti

Come visto nel Capitolo 6, i record nei file ad accesso casuale sono organizzati differentemente da quelli nei file sequenziali: ciascun record ad accesso casuale viene definito con una lunghezza fissa ed inoltre ciascun campo all'interno di un record viene egualmente definito con una lunghezza fissa; la lunghezza prefissata determina la posizione iniziale e finale dei record e dei campi, che non sono separati da virgole, così come i record non sono separati da ritorno carrello ed avanzamento linea.

I file sequenziali salvano i numeri come sequenze di caratteri ASCII, nelle quali ciascun carattere rappresenta una cifra del numero, mentre i file ad accesso casuale salvano i numeri in un formato binario compresso, risparmiando spazio sul disco e tempo di conversione: gli interi memorizzati nei file ad accesso casuale occupano due byte, gli interi lunghi ed i numeri in semplice precisione quattro; i numeri in doppia precisione occupano infine otto byte, mentre le stringhe sono memorizzate come sequenze di caratteri ASCII sia nei file sequenziali che in quelli ad accesso casuale.

8.2 Creazione di un file ad accesso casuale

Un programma che crea un file ad accesso casuale richiede quattro passi:

- definizione dei campi di ciascun record;
- apertura del file in modo ad accesso casuale, specificando la lunghezza di ciascun record;
- assegnazione di dati al record;
- memorizzazione del record nel file.

Per creare un file ad accesso casuale si modificherà STATI ED ABBREVIAZIONI, il programma utilizzato nel Capitolo 7. Se STATI ED ABBREVIAZIONI è stato memorizzato su disco si possono utilizzare i vari metodi di modifica per trasformarlo in un programma che crea un file ad accesso casuale: ne viene presentato di seguito il listato.

```
DECLARE SUB EsaminaFile ()
REM ** STATI ED ABBREVIAZIONI - CREAZIONE E PROVA DEL FILE **
 ' File: UQB0701.BAS

 REM ** Crea il file **
 ' Crea il file StatiAbr.Dat con i nomi degli stati e le abbreviazioni
 ' File sequenziale con due campi: Stato$, Abbr$
```

```
CLS : DEFINT A-Z

OPEN "B:StatiAbr.Dat" FOR OUTPUT AS #1
DO
   READ Stato$, Abbr$            'Legge i dati di un record
   IF Abbr$ = "XX" THEN EXIT DO  'Esce al termine della lettura dei record
   WRITE #1, Stato$, Abbr$       'Scrive un record nel file
LOOP
CLOSE #1

DATA Alabama,AL, Alaska,AK, American Samoa,AS
DATA Arizona,AZ, Arkansas,AR, California,CA
DATA Colorado,CO, Connecticut,CT, Delaware,DE
DATA District of Columbia,DC, Federate States Of Micronesia,TT
DATA Florida,FL, Georgia,GA, Guam,GU
DATA Hawaii,HI, Idaho,ID, Illinois,IL
DATA Indiana,IN, Iowa,IA, Kansas,KS
DATA Kentucky,KY, Louisiana,LA, Maine,ME
DATA Mariana Islands,CM, Marshall Islands,TT, Maryland,MD
DATA Massachusetts,MA, Michigan,MI, Micronesia,TT
DATA Minnesota,MN, Mississippi,MS, Missouri,MO
DATA Montana,MT, Nebraska,NE, Nevada,NV
DATA New Hampshire,NH, New Jersey,NJ, New Mexico,NM
DATA New York,NY, North Carolina,NC, North Dakota,ND
DATA Northern Mariana Island,CM, Ohio,OH, Oklahoma,OK
DATA Oregon,OR, Palau,TT, Pennsylvania,PA
DATA Puerto Rico,PR, Rhode Island,RI, South Carolina,SC
DATA South Dakota,SD, Tennessee,TN, Texas,TX
DATA Utah,UT, Vermont,VT, Virgin Islands,VI
DATA Virginia,VA, Washington,WA, West Virginia,WV
DATA Wisconsin,WI, Wyoming,WY
DATA Fine dei dati,ZZ

REM ** Esamina il file e lo verifica, quindi termina **
CALL EsaminaFile
END
```

L'istruzione TYPE...END TYPE viene utilizzata per definire i record, servendosi contemporaneamente di campi di stringhe e campi numerici: ciò elimina la necessità di funzioni di conversione dei dati in stringhe propria del GW-BASIC e delle prime versioni di QuickBASIC. Il primo passo viene eseguito dalle seguenti linee:

```
'Definizione della struttura TipoRecord
TYPE TipoRecord
  Stato AS STRING * 30              'Campo stringa lungo 30
  Abbr AS STRING * 2                'Campo stringa lungo 2
END TYPE
```

Stato viene definito come un campo contenente una stringa lungo 30 caratteri ed Abbr come un campo contenente una stringa lungo 2, per cui ciascun record sarà lungo 32 caratteri;

questo blocco deve essere aggiunto sotto il blocco "Creazione del file". Il secondo passo richiede l'apertura del file, che era stata compiuta nel Programma 7.1 da:

```
OPEN "B:StatiAbr.Dat" FOR OUTPUT AS #1
```

Aprendo un file ad accesso casuale il programma deve conoscere la lunghezza di ciascun record, che, se non specificata, sarà assunta da QuickBASIC pari a 128 byte. QuickBASIC compie tutti i calcoli necessari in caso di dichiarazione di una variabile di tipo definito, che può essere compiuta tramite le istruzioni DOM, REDIM, COMMON, STATIC o SHARED.

Si utilizzi la clausola LEN = nell'istruzione OPEN per specificare la lunghezza.

```
'Dichiara RecordStato del tipo sopra
DIM RecordStato AS TipoRecord

'Apre il file ad accesso casuale, specifica la lunghezza di un
'record come quella della variabile RecordStato
OPEN "B:StatiAbr.Dat" FOR RANDOM AS #1 LEN = LEN(RecordStato)
```

^calcola la lunghezza del record

Si aggiungano questi blocchi sotto il blocco di dichiarazione del tipo.

Il terzo passo legge i dati per record ed il quarto li memorizza nel file ad accesso casuale; il Programma 7.1 utilizzava il seguente DO...LOOP per un file sequenziale strutturato:

```
DO
  READ Stato$, Abbr$
  IF Abbr$ = "XX" THEN EXIT DO
  WRITE #1, Stato$, Abbr$
LOOP
```

I record dei file ad accesso casuale ricevono un numero al momento della scrittura sul file; nel caso si utilizzi un DO...LOOP per immettere i record e scriverli sul file, tale numero deve essere inizializzato prima di entrare nel ciclo e deve essere incrementato ad ogni ripetizione del DO...LOOP.

I file ad accesso casuale utilizzano l'istruzione PUT # invece di PRINT # e WRITE # per la scrittura dei record. Se i dati sono stati definiti da un'istruzione TYPE...END TYPE, un tipico DO...LOOP per file ad accesso casuale completo di inizializzazione del numero di record potrebbe essere:

```
'Inizializza il numero del record
NumeroRecord = 0
'Legge i dati e li salva nel file
DO
  READ RecordStato.Stato, RecordStato.Abbr
  IF RecordStato.Abbr = "XX" THEN EXIT DO
  NumeroRecord = NumeroRecord + 1
  PUT #1, NumeroRecord, RecordStato
LOOP
```

Si inserisca questo blocco subito dopo quello di apertura del file. I record sono letti un campo alla volta:

```
READ RecordStato.Stato, RecordStato.Abbr
```

^nome record ^nome campo ^separatore

Nel Programma 7.1 il file dei dati veniva aperto in scrittura per poter scrivere i dati su di esso, dopodiché veniva chiuso e riaperto in lettura prima dell'esame dei suoi contenuti.

```
CLOSE #1                                  <nel programma principale
  .
  .
  .
OPEN "B:StatiAbr.Dat" FOR INPUT AS #1     <nel sottoprogramma
```

I file ad accesso casuale permettono di scrivere oppure leggere dai file aperti, senza alcuna necessità di chiudere il file e riaprirlo nel sottoprogramma; perciò si cancelli l'istruzione CLOSE #1 successiva al DO...LOOP del Programma 7.1 e la successiva istruzione OPEN del sottoprogramma EsaminaFile.

Nel programma principale:

```
DO
  READ Stato$, Abbr$            'Legge il dato di un record
  IF Abbr$ = "XX" THEN EXIT DO  'Uscita alla fine dei data
  WRITE #1 Stato$, Abbr$        'Scrive un record nel file
LOOP
                <CLOSE #1 cancellato
DATA . . . . . . . . . . .
  .
  .
  .
```

Nel sottoprogramma:

```
CLS 2
                <commento ed istruzione OPEN cancellati

'Legge un record ad ogni pressione di tasto diverso da U
DO UNTIL EOF(1)
  .
  .
  .
```

Si deve inoltre modificare il blocco di apertura del programma principale in modo che il suo funzionamento si adegui a quanto riportato in questo capitolo; a questo punto la porzione di programma per i file ad accesso casuale è:

```
DECLARE SUB EsaminaFile ()
REM ** STATI ED ABBREVIAZIONI - ACCESSO CASUALE **
' Capitolo 8
```

```
' Microsoft QuickBASIC 4.5  File: UQB0701.BAS

REM ** Crea il file **
' Crea il file StatiAbr.Dat con i nomi degli stati e le abbreviazioni
' File sequenziale con due campi: Stato$, Abbr$
CLS : DEFINT A-Z

'Definizione della struttura TipoRecord
TYPE TipoRecord
   Stato AS STRING * 30                'Campo stringa lungo 30
   Abbr AS STRING * 2                  'Campo stringa lungo 2
END TYPE

'Dichiara RecordStato del tipo sopra
DIM RecordStato AS TipoRecord

'Apre il file ad accesso casuale, specifica la lunghezza di un
'record come quella della variabile RecordStato
OPEN "B:StatiAbr.Dat" FOR RANDOM AS #1 LEN = LEN(RecordStato)

'Inizializza il numero del record
NumeroRecord = 0

'Legge i dati e li salva nel file
DO
   READ RecordStato.Stato, RecordStato.Abbr
   IF RecordStato.Abbr = "XX" THEN EXIT DO
   NumeroRecord = NumeroRecord + 1
   PUT #1, NumeroRecord, RecordStato
LOOP

DATA Alabama,AL, Alaska,AK, American Samoa,AS
DATA Arizona,AZ, Arkansas,AR, California,CA
DATA Colorado,CO, Connecticut,CT, Delaware,DE
DATA District of Columbia,DC, Federate States Of Micronesia,TT
DATA Florida,FL, Georgia,GA, Guam,GU
DATA Hawaii,HI, Idaho,ID, Illinois,IL
DATA Indiana,IN, Iowa,IA, Kansas,KS
DATA Kentucky,KY, Louisiana,LA, Maine,ME
DATA Mariana Islands,CM, Marshall Islands,TT, Maryland,MD
DATA Massachusetts,MA, Michigan,MI, Micronesia,TT
DATA Minnesota,MN, Mississippi,MS, Missouri,MO
DATA Montana,MT, Nebraska,NE, Nevada,NV
DATA New Hampshire,NH, New Jersey,NJ, New Mexico,NM
DATA New York,NY, North Carolina,NC, North Dakota,ND
DATA Northern Mariana Island,CM, Ohio,OH, Oklahoma,OK
DATA Oregon,OR, Palau,TT, Pennsylvania,PA
DATA Puerto Rico,PR, Rhode Island,RI, South Carolina,SC
DATA South Dakota,SD, Tennessee,TN, Texas,TX
```

```
DATA Utah,UT, Vermont,VT, Virgin Islands,VI
DATA Virginia,VA, Washington,WA, West Virginia,WV
DATA Wisconsin,WI, Wyoming,WY
DATA Fine dei dati,ZZ

REM ** Esamina il file e lo verifica, quindi termina **
CALL EsaminaFile
END

SUB EsaminaFile STATIC
' Esamina il file StatiAbr.Dat, un record alla volta

  ' Scrive le istruzioni nelle linee da 21 a 25
  Text$(1) = STRING$(74, 196)
  Text$(2) = "Esamina il file degli Stati e delle Abbreviazioni."
  Text$(3) = "Porre il disco dei dati nel drive B."
  Text$(4) = "Premere un tasto per iniziare: apparira' un record."
  Text$(5) = "Premere la barra spaziatrice per avanzare, U per uscire."
  CLS
  FOR linea = 1 TO 5
    LOCATE linea + 20, 3: PRINT Text$(linea);
  NEXT linea

  ' Definisce le linee da 1 a 20 come finestra di visualizzazione dei record
  VIEW PRINT 1 TO 20

  ' Attende la pressione di un tasto per iniziare

  PRINT "Premere un tasto per iniziare."
  untasto$ = INPUT$(1)
  CLS 2                          'Cancella la finestra di visualizzazione

  ' Apre in input il file StatiAbr.Dat del drive B con il numero 1
  OPEN "B:StatiAbr.Dat" FOR INPUT AS #1

  ' Legge un record ad ogni pressione di tasto diverso da U
  DO UNTIL EOF(1)
    LINE INPUT #1, record$
    PRINT record$
    PRINT
    prossimo$ = INPUT$(1)
    IF UCASE$(prossimo$) = "U" THEN EXIT DO
  LOOP

  ' Chiude il file e termina il sottoprogramma
  VIEW PRINT 1 TO 25: CLS
  CLOSE #1

END SUB
```

8.3 Lettura di un file ad accesso casuale

Il programma STATI ED ABBREVIAZIONI del Capitolo 7 chiamava un sottoprogramma per esaminare i dati del file sequenziale strutturato.

```
REM ** Esamina il file per verificarne i contenuti **
CALL EsaminaFile
```

Queste due linee possono anche essere utilizzate per un file ad accesso casuale, ma si dovrebbe cambiare il nome del sottoprogramma per evitare confusioni (ad esempio in LeggiFile).

```
REM ** Esamina il file per verificarne i contenuti **
CALL LeggiFile
```

Il sottoprogramma utilizzato in STATI ED ABBREVIAZIONI può venire modificato per funzionare con file ad accesso casuale cambiando poche linee di codice: ci si assicuri di modificare il nome del sottoprogramma per conformarsi a quello utilizzato nella chiamata dal programma principale.

```
SUB LeggiFile
```

L'istruzione OPEN del Programma 7.1 può essere cancellata (come precedentemente indicato) insieme al commento che la precede poiché nella versione ad accesso casuale il file è ancora aperto al momento della chiamata del sottoprogramma.

I tipi definiti dall'utente devono essere dichiarati in un'apposito blocco di dichiarazione TYPE...END TYPE prima di essere utilizzati nel programma; sebbene un tipo definito dall'utente possa essere dichiarato solamente nel codice a livello di modulo, si possono dichiarare variabili di un tipo definito dall'utente (come RecordStato) in qualsiasi punto del modulo, anche in una SUBroutine od una FUNCTION, servendosi di un'istruzione DIM, REDIM, COMMON, STATIC o SHARED per assegnare ad un nome di variabile un tipo definito dall'utente.

Per dichiarare che RecordStato è definita nello stesso modo della struttura TYPE...END TYPE del programma principale, deve essere aggiunta un'istruzione DIM nel sottoprogramma immediatamente dopo l'istruzione CLS 2.

```
untasto$ = INPUT$(1)
CLS 2

' Dichiara RecordStato come definito nel programma principale
DIM RecordStato AS TipoRecord
```

STATI ED ABBREVIAZIONI utilizzava il seguente DO...LOOP per leggere i record del file strutturato sequenzialmente:

```
DO UNTIL EOF(1)
  LINE INPUT #1, record$
```

```
  PRINT record$
  PRINT
  prossimo$ = INPUT$(1)
  IF UCASE$(prossimo$) = "U" THEN EXIT DO
 LOOP
```

I file ad accesso casuale utilizzano l'istruzione GET # per leggere i record, per cui il DO...LOOP deve essere cambiato in:

```
NumeroRecord = 1
DO UNTIL EOF(1)
  GET #1, NumeroRecord, RecordStato
  PRINT RecordStato.Stato; RecordStato.Abbr
  NumeroRecord = NumeroRecord + 1
  prossimo$ = INPUT$(1)
  IF UCASE$(prossimo$) = "U" THEN EXIT DO
 LOOP
```

Con queste modifiche viene completato il Programma 8.1, STATI ED ABBREVIAZIONI - ACCESSO CASUALE.

```
DECLARE SUB LeggiFile ()
REM ** STATI ED ABBREVIAZIONI - ACCESSO CASUALE **
' Capitolo 8
' Microsoft QuickBASIC 4.5  File: UQB0701.BAS

REM ** Crea il file **
' Crea il file StatiAbr.Dat con i nomi degli stati e le abbreviazioni
' File sequenziale con due campi: Stato$, Abbr$
CLS : DEFINT A-Z

'Definizione della struttura TipoRecord
TYPE TipoRecord
   Stato AS STRING * 30                'Campo stringa lungo 30
   Abbr AS STRING * 2                  'Campo stringa lungo 2
END TYPE

'Dichiara RecordStato del tipo sopra
DIM RecordStato AS TipoRecord

'Apre il file ad accesso casuale, specifica la lunghezza di un
'record come quella della variabile RecordStato
OPEN "B:StatiAbr.Dat" FOR RANDOM AS #1 LEN = LEN(RecordStato)

'Inizializza il numero del record
NumeroRecord = 0

'Legge i dati e li salva nel file
DO
   READ RecordStato.Stato, RecordStato.Abbr
```

```
   IF RecordStato.Abbr = "XX" THEN EXIT DO
   NumeroRecord = NumeroRecord + 1
   PUT #1, NumeroRecord, RecordStato
LOOP

DATA Alabama,AL, Alaska,AK, American Samoa,AS
DATA Arizona,AZ, Arkansas,AR, California,CA
DATA Colorado,CO, Connecticut,CT, Delaware,DE
DATA District of Columbia,DC, Federate States Of Micronesia,TT
DATA Florida,FL, Georgia,GA, Guam,GU
DATA Hawaii,HI, Idaho,ID, Illinois,IL
DATA Indiana,IN, Iowa,IA, Kansas,KS
DATA Kentucky,KY, Louisiana,LA, Maine,ME
DATA Mariana Islands,CM, Marshall Islands,TT, Maryland,MD
DATA Massachusetts,MA, Michigan,MI, Micronesia,TT
DATA Minnesota,MN, Mississippi,MS, Missouri,MO
DATA Montana,MT, Nebraska,NE, Nevada,NV
DATA New Hampshire,NH, New Jersey,NJ, New Mexico,NM
DATA New York,NY, North Carolina,NC, North Dakota,ND
DATA Northern Mariana Island,CM, Ohio,OH, Oklahoma,OK
DATA Oregon,OR, Palau,TT, Pennsylvania,PA
DATA Puerto Rico,PR, Rhode Island,RI, South Carolina,SC
DATA South Dakota,SD, Tennessee,TN, Texas,TX
DATA Utah,UT, Vermont,VT, Virgin Islands,VI
DATA Virginia,VA, Washington,WA, West Virginia,WV
DATA Wisconsin,WI, Wyoming,WY
DATA Fine dei dati,XX

REM ** Esamina il file e lo verifica, quindi termina **
CALL EsaminaFile
END

SUB LeggiFile
' Esamina il file StatiAbr.Dat, un record alla volta

   ' Scrive le istruzioni nelle linee da 21 a 25
   Text$(1) = STRING$(74, 196)
   Text$(2) = "Esamina il file degli Stati e delle Abbreviazioni."
   Text$(3) = "Porre il disco dei dati nel drive B."
   Text$(4) = "Premere un tasto per iniziare: apparira' un record."
   Text$(5) = "Premere la barra spaziatrice per avanzare, U per uscire."
   CLS
   FOR linea = 1 TO 5
     LOCATE linea + 20, 3: PRINT Text$(linea);
   NEXT linea

   ' Definisce le linee da 1 a 20 come finestra di visualizzazione dei record
   VIEW PRINT 1 TO 20
   ' Attende la pressione di un tasto per iniziare
```

```
  PRINT "Premere un tasto per iniziare."
  untasto$ = INPUT$(1)
  CLS 2                          'Cancella la finestra di visualizzazione
  ' Apre in input il file StatiAbr.Dat del drive B con il numero 1
  OPEN "B:StatiAbr.Dat" FOR INPUT AS #1

  NumeroRecord = 1
  DO UNTIL EOF(1)
   GET #1, NumeroRecord, RecordStato
   PRINT RecordStato.Stato; RecordStato.Abbr
   NumeroRecord = NumeroRecord + 1
   prossimo$ = INPUT$(1)
   IF UCASE$(prossimo$) = "U" THEN EXIT DO
  LOOP
  ' Chiude il file e termina il sottoprogramma
  VIEW PRINT 1 TO 25: CLS
  CLOSE #1
END SUB
```

Programma 8.1 STATI ED ABBREVIAZIONI - ACCESSO CASUALE

La Figura 8.1 mostra un'esempio di schermata prodotta dal programma quando l'istruzione PRINT include un punto e virgola tra i campi stato ed abbreviazione:

```
PRINT RecordStato.Stato; RecordStato.Abbr


American Samoa                  AS
Arizona                         AZ
Arkansas                        AR
California                      CA
Colorado                        CO
Connecticut                     CT
Delaware                        DE
District of Columbia            DC
Federated States of Micronesia  TT


  Esamina il file degli stati e delle abbreviazioni.
  Porre il disco dei dati nel drive B.
  Premere un tasto per iniziare: apparira' un record.
  Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 8.1 Output del Programma 8.1 utilizzando la spaziatura con punto e virgola

La spaziatura appare identica a quella dei dati nel file ad accesso casuale: trenta byte sono utilizzati per memorizzare lo stato e due per l'abbreviazione. È possibile ricercare particolari elementi nel file, dal momento che la lunghezze dei record e dei loro campi è costante.

La Figura 8.2 illustra la schermata di output nel caso l'istruzione PRINT contenga "TAB(40)" tra i campi stato ed abbreviazione:

```
PRINT RecordStato.Stato; TAB(40); RecordStato.Abbr
```

Come appare chiaro osservando quanto riportato dalla Figura 8.2, sebbene i record non siano memorizzati nel file in questo formato, la schermata appare meno costretta grazie alla spaziatura tra i due campi fornita dalla function TAB al momento della scrittura sullo schermo.

```
American Samoa                    AS
Arizona                           AZ
Arkansas                          AR
California                        CA
Colorado                          CO
Connecticut                       CT
Delaware                          DE
District of Columbia              DC
Federated States of Micronesia    TT

  Esamina il file degli stati e delle abbreviazioni.
  Porre il disco dei dati nel drive B.
  Premere un tasto per iniziare: apparira' un record.
  Premere la barra spaziatrice per avanzare, U per uscire.
```

Figura 8.2 Output del Programma 8.1 utilizzando TAB(40) per la spaziatura

8.4 Esempio: valore nutrizionale degli alimenti

Il prossimo progetto di programmazione è basato su un file dati ad accesso casuale ed include:

- un programma principale con un menu che consente la selezione di un oggetto da una lista;
- un sottoprogramma che aggiunge record ad un file;
- un sottoprogramma che legge tutti i record del file dati;
- un sottoprogramma che legge record selezionati dal numero di record nel file dati;
- un sottoprogramma di ricerca di record specifici in base ad una chiave.

I dati del file sono ripresi da un trattato sul valore nutrizionale dei principali cibi edito dalla Dover Publications per il dipartimento dell'agricoltura degli Stati Uniti, sebbene nel file sia utilizzata solo una piccola parte della base di dati disponibile in tale pubblicazione; in ogni caso si possono aggiungere nuovi dati al file in qualsiasi momento. Le informazioni comprendono i nomi dei vari cibi, le loro condizioni (fresco, crudo, cotto e così via) ed il numero di calorie per oncia (1 oncia = 28,35 grammi); a ciascuna di queste categorie corrisponde un campo dei record del file ad accesso casuale dei dati.

Il programma verrà realizzato in moduli corti, che potranno essere controllati separatamente prima di scrivere le parti successive. Si cominci digitando il programma che segue ed eseguendolo: esso richiede di indicare il drive da utilizzare per memorizzare il file ed il relativo nome; le due informazioni vengono riunite per formare il nome completo del file (ciò rende questo programma adatto all'uso con altri file dati).

```
REM ** Valori nutrizionali **
' Capitolo 8
' Microsoft QuickBASIC  4.5 File: UQB0802.BAS
REM ** Richiede il drive ed il nome del file **
CLS : DEFINT A-Z
LOCATE 5, 5: INPUT "Drive contenente il file: "; drive$
drive$ = LEFT$(drive$, 1)
LOCATE 7, 5: INPUT "Nome del file con estensione: "; nomefile$
nomefile$ = drive$ + ":" + nomefile$
END
```

Una tipica risposta da dare a questa parte di programma è:

```
Drive contenente il file: ?B
Nome del file con estensione:  ?FileAlimenti.Dat
```

A questo punto occorre aggiungere una sezione che dichiari la struttura del record nel programma principale (non in un sottoprogramma o nel corpo di una function); il frammento deve essere posto immediatamente sopra all'istruzione END del primo segmento.

```
REM ** Definisce la struttura del record **
TYPE TipoAlimento
  alimento AS STRING * 18
  tipo AS STRING * 15
```

```
  calorie AS INTEGER
END TYPE
END
```

Questa sezione definisce un record con tre campi, la cui lunghezza totale è 35 byte: alimento è una campo contenente una stringa lungo 18 byte, tipo è un campo contenente una stringa lungo 15 byte e calorie è un intero da 2 byte.

Si aggiunga alla porzione di programma precedente la sezione riportata di seguito, che fa comparire un menu sopra l'istruzione END dell'ultima parte del programma.

```
REM ** Scrive il menu **
DO
  LOCATE 5, 5: PRINT SPACE$(70); : LOCATE 7, 5: PRINT SPACE$(70);
  LOCATE 3, 23: PRINT STRING$(35, 220);
  LOCATE 19, 23: PRINT STRING$(35, 223);
  FOR riga = 4 TO 18
    LOCATE riga, 23: PRINT CHR$(221);
    LOCATE riga, 57: PRINT CHR$(222);
  NEXT riga
  LOCATE 5, 29: PRINT "MENU FILE AD ACCESSO CASUALE"
  LOCATE 9, 29: PRINT "(A)ggiunge record";
  LOCATE 11, 29: PRINT "(L)egge tutti i record";
  LOCATE 13, 29: PRINT "(T)rova un record specifico";
  LOCATE 15, 29: PRINT "(R)icerca record";
  LOCATE 17, 29: PRINT "(U)scita";
LOOP
END
```

Si esegua il segmento di programma così ottenuto: dopo avere digitato il drive ed il nome del file da utilizzare apparirà sullo schermo il menu, come mostrato nella Figura 8.3. Dal momento che a questo punto il DO...LOOP non è ancora completo, si prema ESC per uscire.

La prossima parte, presentata qui di seguito, consente di selezionare una delle opzioni del menu ed è implementata da un'istruzione SELECT CASE che effettua una chiamata al sottoprogramma appropriato, scelto in base al tasto premuto dall'utente; si noti che un'istruzione BEEP viene utilizzata per emettere un suono che indica l'attesa della pressione di un tasto.

```
REM ** Compie la scelta **
LOCATE 22, 3: PRINT SPACE$(70);
LOCATE 22, 3: PRINT "Digitare un'opzione (A, L, T, R, od U)";
DO
  BEEP: TastoScelto$ = UCASE$(INPUT$(1))
LOOP WHILE INSTR("ALTRU", TastoScelto$) = 0
```

Nel caso si intenda provare il programma parziale appena realizzato, si ricordi che le procedure chiamate dalla SELECT CASE non sono ancora state scritte: a scopo di verifica si pongano delle istruzioni PRINT vicino alla SELECT CASE dopo avere racchiuso tra virgolette le chiamate ai sottoprogrammi.

```
MENU FILE AD ACCESSO CASUALE

(A)ggiunge record
(L)egge tutti i record
(T)rova un record specifico
(R)icerca record
(U)scita

Digitare un'opzione (A, L, T, R od U)
```

Figura 8.3 Menu del Programma 8.2

```
SELECT CASE TastoScelto$
  CASE "A"
    CALL AggiungiRecord(nomefile$)
  CASE "L"
    PRINT "CALL LeggiRecord(nomefile$)"
  CASE "T"
    PRINT "CALL TrovaRecord(nomefile$)"
  CASE "R"
    PRINT "CALL RicercaRecord(nomefile$)"
  CASE "U"
    EXIT DO
END SELECT
```

Si aggiunga questa sezione dopo il blocco di stampa del menu:

```
REM ** Compie la scelta **
LOCATE 22, 3: PRINT SPACE$(70);
LOCATE 22, 3: PRINT "Digitare un'opzione (A, L, T, R, od U)";
DO
  BEEP: TastoScelto$ = UCASE$(INPUT$(1))
LOOP WHILE INSTR("ALTRU", TastoScelto$) = 0
SELECT CASE TastoScelto$
  CASE "A"
    CALL AggiungiRecord(nomefile$)
  CASE "L"
    PRINT "CALL LeggiRecord(nomefile$)"
  CASE "T"
    PRINT "CALL TrovaRecord(nomefile$)"
```

```
    CASE "R"
      PRINT "CALL RicercaRecord(nomefile$)"
    CASE "U"
      EXIT DO
  END SELECT
  CLS
LOOP
CLS : END
```

Eseguendo questo segmento di programma appariranno i seguenti prompt:

```
Drive contenente il file:  ?B
Nome del file con estensione:  ?FileAlimenti.Dat
```

Si premano i tasti corrispondenti alle prime lettere delle varie opzioni del menu per controllare la sezione con l'istruzione SELECT CASE; una volta apparso il messaggio corrispondente alla scelta compiuta, si prema un tasto per fare ricomparire il menu ed effettuare una nuova selezione.

Il compito successivo è aggiungere le procedure necessarie ad eseguire le opzioni della SELECT CASE; serviranno le procedure per i casi seguenti.

Caso "A"	Sottoprogramma AggiungiRecord
Caso "L"	Sottoprogramma LeggiRecord
Caso "T"	Sottoprogramma TrovaRecord
Caso "R"	Sottoprogramma RicercaRecord

8.4.1 Sottoprogramma AggiungiRecord

Il sottoprogramma AggiungiRecord, immesso con il comando Nuova SUBroutine del menu Modifica può essere utilizzato per creare il file dati originale od aggiungere record ad un file esistente. Digitando AggiungiRecord come nome del sottoprogramma appariranno le linee di apertura e chiusura del sottoprogramma:

```
SUB AggiungiRecord_
END SUB
```

Si aggiungano un'istruzione CLS ed un'istruzione DIM per dichiarare una variabile (RecordAlimento) utilizzando il tipo di dati definito nel programma principale.

```
SUB AggiungiRecord (nomefile$)
```

```
  CLS
  ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento
```

Quindi viene aperto il file.

```
  'Apre un file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)
```

Viene calcolato il numero di record già presenti nel file, se ve ne sono.

```
  ' Utilizza LOF(1) per calcolare i record gia' nel file
  NumeroRecord = LOF(1) \ LEN(RecordAlimento)
```

La function LOF(1) calcola il numero complessivo di byte nel file con la divisione intera (\); se il file dati è nuovo non contiene record e LOF(1) restituisce zero in questo caso; LEN(RecordAlimento) calcola il numero di byte in un record, per cui:

NumeroRecord = Byte complessivi nel file / Byte in un record

Dal momento che i record di un file ad accesso casuale hanno tutti la stessa dimensione, questa formula può essere sempre utilizzata per calcolare il numero di record di un file.

Vengono ora immessi i nuovi record e chiuso il file quando finito; il tutto viene svolto dal segmento di programma seguente:

```
  ' Aggiunge i record e chiude il file
  DO
    INPUT "Nome dell'alimento ?", RecordAlimento.alimento
    INPUT "Tipo dell'alimento ?", RecordAlimento.tipo
    INPUT "Calorie per oncia ?", RecordAlimento.calorie
    NumeroRecord = NumeroRecord + 1
    PUT #1, NumeroRecord, RecordAlimento
    INPUT "Altro (S o N) ? ", ancora$
    PRINT
  LOOP UNTIL UCASE$(ancora$) = "N"
  CLOSE #1
END SUB
```

Il Programma 8.2, FILE VALORI NUTRIZIONALI PRELIMINARE, contiene a questo punto il programma principale completo ed un sottoprogramma (AggiungiRecord).

```
DECLARE SUB AggiungiRecord (nomefile$)
REM ** VALORI NUTRIZIONALI PRELIMINARE **
' Capitolo 8
' Microsoft QuickBASIC  4.5 File: UQB0802.BAS

REM ** Richiede il drive ed il nome del file **
CLS : DEFINT A-Z
LOCATE 5, 5: INPUT "Drive contenente il file: "; drive$
drive$ = LEFT$(drive$, 1)
LOCATE 7, 5: INPUT "Nome del file con estensione: "; nomefile$
nomefile$ = drive$ + ":" + nomefile$
```

```
REM ** Definisce la struttura del record **
TYPE TipoAlimento
  alimento AS STRING * 18
  tipo AS STRING * 15
  calorie AS INTEGER
END TYPE

REM ** Scrive il menu **
DO
  LOCATE 5, 5: PRINT SPACE$(70); : LOCATE 7, 5: PRINT SPACE$(70);
  LOCATE 3, 23: PRINT STRING$(35, 220);
  LOCATE 19, 23: PRINT STRING$(35, 223);
  FOR riga = 4 TO 18
    LOCATE riga, 23: PRINT CHR$(221);
    LOCATE riga, 57: PRINT CHR$(222);
  NEXT riga
  LOCATE 5, 29: PRINT "MENU FILE AD ACCESSO CASUALE"
  LOCATE 9, 29: PRINT "(A)ggiunge record";
  LOCATE 11, 29: PRINT "(L)egge tutti i record";
  LOCATE 13, 29: PRINT "(T)rova un record specifico";
  LOCATE 15, 29: PRINT "(R)icerca record";
  LOCATE 17, 29: PRINT "(U)scita";

  REM ** Compie la scelta **
  LOCATE 22, 3: PRINT SPACE$(70);
  LOCATE 22, 3: PRINT "Digitare un'opzione (A, L, T, R, od U)";
  DO

    BEEP: TastoScelto$ = UCASE$(INPUT$(1))
  LOOP WHILE INSTR("ALTRU", TastoScelto$) = 0
  SELECT CASE TastoScelto$
    CASE "A"
      CALL AggiungiRecord(nomefile$)
    CASE "L"
      PRINT "CALL LeggiRecord(nomefile$)"
    CASE "T"
      PRINT "CALL TrovaRecord(nomefile$)"
    CASE "R"
      PRINT "CALL RicercaRecord(nomefile$)"
    CASE "U"
      EXIT DO
  END SELECT
  CLS
LOOP
CLS : END

SUB AggiungiRecord (nomefile$)
  CLS
  ' Dichiara il tipo di RecordAlimento
```

```
  DIM RecordAlimento AS TipoAlimento

  ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)

  ' Calcola il numero di record con LOF
  NumeroRecord = LOF(1) \ LEN(RecordAlimento)

  ' Aggiunge i record e chiude il file
  DO
    INPUT "Nome dell'alimento ?", RecordAlimento.alimento
    INPUT "Tipo dell'alimento ?", RecordAlimento.tipo
    INPUT "Calorie per oncia ?", RecordAlimento.calorie
    NumeroRecord = NumeroRecord + 1
    PUT #1, NumeroRecord, RecordAlimento
    INPUT "Altro (S o N) ? ", ancora$
    PRINT
  LOOP UNTIL UCASE$(ancora$) = "N"
  CLOSE #1
END SUB
END
```

Programma 8.2 FILE VALORI NUTRIZIONALI PRELIMINARE

Per verificare il sottoprogramma appena digitato corrispondente al caso A si ritorni al programma principale, si rimuova la parola chiave PRINT e le virgolette in modo che la chiamata al sottoprogramma appaia come segue:

```
CASE "A"
  CALL AggiungiRecord(nomefile$)
```

Si esegua il programma e si selezioni (A)ggiungi record dal menu mostrato nella Figura 8.3 premendo **a**: lo schermo verrà pulito ed apparirà il prompt d'apertura:

```
Nome dell'alimento?_
```

Si digitino i dati di tre record come mostrato nella Figura 8.4 ed una volta finito si prema n in risposta all'ultimo prompt:

```
Altro (S o N) ? n
```

Ciò riporterà al menu: si prema **q** per uscire e si tornerà al programma nella finestra di visualizzazione.

```
Nome dell'alimento ?Avocado
Tipo dell'alimento ?fresco
```

```
Calorie per oncia ?48
Altro (S o N) ? s

Nome dell'alimento ?Banana
Tipo dell'alimento ?fresca
Calorie per oncia ?24
Altro (S o N) ? s

Nome dell'alimento ?Biscotti
Tipo dell'alimento ?fatti in casa
Calorie per oncia ?105
Altro (S o N) ? n
```

Figura 8.4 Tre record per il file degli alimenti

8.4.2 Sottoprogramma LeggiRecord

Dopo avere introdotto tre record nel file dei dati dei cibi è necessario avere qualche metodo per leggere i record. Il sottoprogramma LeggiRecord è simile ad AggiungiRecord, è dichiarato il tipo RecordAlimento e viene aperto il file nello stesso modo, quindi LOF(1)è utilizzata per calcolare il numero complessivo di record nel file, come segue:

```
NumeroDiRecord = LOF(1) \ LEN(RecordAlimento)
```

Il sottoprogramma utilizza un ciclo FOR...NEXT per leggere i record e visualizzarli:

```
FOR NumeroRecord = 1 TO NumeroDiRecord
  GET #1, NumeroRecord, RecordAlimento
  PRINT RecordAlimento.alimento; RoodRecord.tipo;
  PRINT RecordAlimento.calorie
NEXT NumeroRecord
```

Il file viene quindi chiuso ed il sottoprogramma termina come prima. Il sottoprogramma completo è:

```
SUB LeggiRecord (nomefile$)
  CLS
   ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento
  ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)
  ' Calcola il numero di record con LOF
```

```
  NumeroRecord = LOF(1) \ LEN(RecordAlimento)
  ' Legge e visualizza i record
  FOR NumeroRecord = 1 TO NumeroDiRecord
    GET #1, NumeroRecord, RecordAlimento
    PRINT RecordAlimento.alimento; RoodRecord.tipo;
    PRINT RecordAlimento.calorie
  NEXT NumeroRecord

  LOCATE 25, 1: PRINT "Premere un tasto per continuare";
  attesa$ = INPUT$(1)
  CLOSE #1
END SUB
```

Si immetta il sottoprogramma per mezzo del comando Nuova SUBroutine del menu Modifica, servendosi del nome LeggiRecord. Prima di potere verificare il sottoprogramma LeggiRecord si deve cancellare la parola chiave PRINT e le virgolette dalla sezione SELECT CASE corrispondente del programma principale.

```
SELECT CASE TastoScelto$
  CASE "A"
    CALL AggiungiRecord(nomefile$)
  CASE "L"
    CALL LeggiRecord(nomefile$)
  CASE "T"
    PRINT "CALL TrovaRecord(nomefile$)"    <eliminare PRINT e virgolette
  CASE "R"
    PRINT "CALL RicercaRecord(nomefile$)"
```

Si esegua il Programma 8.2 con in più questo sottoprogramma e si prema l per selezionare (L)eggi record dal menu del programma.

Precedentemente sono stati introdotti tre record al momento della creazione del file: la Figura 8.5 mostra i tre record mostrati sul video dal sottoprogramma LeggiRecord

Avocado	**fresco**	**48**
Banana	**fresca**	**24**
Biscotti	**fatti in casa**	**105**

Figura 8.5 Tre record del file degli alimenti

8.4.3 Scelta di un record per numero

I record dei file ad accesso casuale sono numerati dall'istruzione PUT#: per questo è molto semplice recuperare un record da un file dati utilizzando l'istruzione GET #. Il sottopro-

gramma TrovaRecord mostrato sotto consente di recuperare un record specifico immettendo il suo numero: questa funzione non sarà probabilmente utilizzata spesso, ma viene presentata per illustrare il suo funzionamento.

```
SUB TrovaRecord (nomefile$)
  CONST Falso = 0, Vero = NOT Falso
  CLS

  ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento
   ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)
   ' Calcola il numero di record con LOF
  NumeroRecord = LOF(1) \ LEN(RecordAlimento)
   ' Trova il record richiesto
  LeggeAltri = Vero
  DO
    PRINT "Numero del record: ";
    INPUT "", NumeroRecord
    IF NumeroRecord > 0 AND NumeroRecord <= NumeroDiRecord THEN
      GET #1, NumeroRecord, RecordAlimento
      PRINT RecordAlimento.alimento; RecordAlimento.tipo;
      PRINT RecordAlimento.calorie
    ELSEIF NumeroRecord = 0 THEN
      LeggeAltri = Falso
    ELSE
      PRINT "Valore fuori dai limiti."
    END IF
    PRINT
  LOOP WHILE LeggeAltri
  LOCATE 25, 1: PRINT "Premere un tasto per continuare";
  attesa$ = INPUT$(1)
  CLOSE #1
END SUB
```

All'inizio del sottoprogramma vengono dichiarate due costanti, Vero e Falso:

```
CONST Falso = 0, Vero = NOT Falso
```

Viene utilizzato un DO...LOOP per richiedere il numero del record, leggerlo e visualizzare le informazioni corrispondenti. La variabile LeggeAltri riceve il valore Vero prima di entrare nel ciclo, all'interno del quale viene richiesto all'utente un numero di record; il comportamento successivo alla risposta può essere diverso:

- se il numero è nel campo dei numeri dei record del file viene visualizzato il record corrispondente;

```
IF NumeroRecord > 0 AND NumeroRecord <= NumeroDiRecord THEN
GET #1, NumeroRecord, RecordAlimento
PRINT RecordAlimento.alimento; RecordAlimento.tipo;
PRINT RecordAlimento.calorie
```

se il numero è zero LeggiAltri viene posta uguale a Falso, causando l'uscita dal ciclo.

```
        ELSEIF NumeroRecord = 0 THEN
    LeggeAltri = Falso
    .
    .
    .
    LOOP WHILE LeggiAltri                <loop se LeggiAltri e' Vero
```

- se il numero è maggiore del numero di record nel file o minore di zero viene mostrato un messaggio di valore fuori dai limiti e il ciclo viene ripetuto per consentire un'altra scelta.

```
    ELSE
      PRINT "Valore fuori dai limiti."
    END IF
    PRINT
    LOOP WHILE LeggeAltri
    END SUB
```

Il sottoprogramma termina quando si esce dal ciclo scrivendo zero alla richiesta del numero di record.

Prima di provare l'opzione Trova record del menu, si lanci nuovamente il programma e si aggiungano i record mostrati nella Tabella 8.1 servendosi della selezione Aggiungi record; una volta completato il record del pane degli angeli si scriva **n** al prompt:

```
Ancora (S o N) ? n
```

Tabella 8.1 Valori nutrizionali aggiuntivi

Alimento	Tipo	Calorie
Avogado	fresco	105
Banana	fresca	24
Biscotti	casalinghi	105
Branzino	alla griglia	74
Broccoli	cotti	7
Burro	fritto	205
Caciotta	stagionata	30
Carne	ai ferri	13
Carote	crude	12
Fagioli	lessati	32
Insalata	cruda	7
Mirtilli	freschi	18

Alimento	Tipo	Calorie
Mela	fresca	15
Pane	integrale	64
Pollo	arrosto	63
Torta	meringata	105
Torta	pane degli angeli	77

Viene mostrato nuovamente il menu; il file ora contiene 17 record che possono essere mostrati tutti insieme. Tuttavia occorre rimuovere la parola chiave PRINT e le virgolette dalla sezione SELECT CASE corrispondente del programma principale per potere utilizzare il sottoprogramma TrovaRecord.

```
SELECT CASE TastoScelto$
  CASE "A"
    CALL AggiungiRecord(nomefile$)
  CASE "L"
    CALL LeggiRecord(nomefile$)
  CASE "T"
    CALL TrovaRecord(nomefile$)
  CASE "R"
    PRINT "CALL RicercaRecord(nomefile$)"
        .
        .
        .
```

Si selezioni LeggiRecord dal menu ed apparirà la schermata mostrata nella Figura 8.6; si prema un tasto per ritornare al menu del programma, quindi si selezioni TrovaRecord; al prompt si digiti il numero del record che si desidera vedere e si prema **INVIO**: verrà mostrato il record richiesto, insieme al prompt che richiede un altro record.

```
Immettere il numero di record desiderato          <nessun cursore
```

La Figura 8.7 mostra le risposte ad alcuni input.

```
Avocado        fresco          48
Banana         fresca          24
Biscotti       casalinghi      105
Branzino       alla griglia    74
Broccoli       cotti           7
Burro          fritto          205
Caciotta       stagionata      30
```

```
Carne                ai ferri            93
Carote               crude               12
Fagioli              lessati             32
Insalata             cruda               7
Mirtilli             freschi             18
Mela                 fresca              15
Pane                 integrale           64
Pollo                arrosto             83
Torta                meringata           105
Torta                pane degli angeli   77


Premere un tasto per continuare
```

Figura 8.6 Output del file dei cibi - 17 record

```
Numero del record: 2
Banana        fresca          24

Numero del record: 4
Branzino      alla griglia    74

Numero del record: 10
Fagioli       lessati         32

Numero del record: 8
Carne         ai ferri        93

Numero del record: 0


Premere un tasto per continuare
```

Figura 8.7 Output di TrovaRecord

8.4.4 Ricerca dei record in base al nome dell'alimento

Probabilmente sarà difficile ricordare i contenuti dei record a partire dal loro numero, per cui sarà gradito sapere che è possibile ricercare i record partendo dal nome dell'alimento che si intende esaminare.

Il sottoprogramma RicercaRecord consente di visualizzare il record completo del file dei dati corrispondente a qualsiasi alimento nominato, purché sia tra quelli presenti nel file. Non è necessario scrivere per intero il nome dell'alimento: la lunghezza della chiave digitata da tastiera viene assegnata alla variabile Lungh e la ricerca viene fatta solo in base alla parte di nome immessa.

```
' Legge la chiave di ricerca
INPUT "Digitare il nome dell'alimento da cercare: "; elem$
Lungh = LEN(elem$)
elem$ = UCASE$(elem$)
```

Se si digita una sola lettera verranno visualizzati tutti i record dei cibi che cominciano con quella lettera

```
Digitare il nome dell'alimento da cercare: ? p
Pane            integrale          69
Pollo           arrosto            83
```

Una volta immessa una stringa da cercare, la ricerca dei cibi le cui lettere iniziali corrispondono a quella stringa inizia dal primo record in ordine numerico e prosegue un record alla volta per tutto il file fino a raggiungere un alimento il cui nome sia superiore alla chiave, cioèalfabeticamente successivo al nome dato, condizione che causa l'uscita dal ciclo; il puntatore al file viene posizionato in corrispondenza del primo record del file dall'istruzione SEEK.

```
'Scrive i record del file corrispondenti
NumeroRecord = 1; trovato = 0
DO UNTIL EOF(1)
  GET #1, NumeroRecord, RecordAlimento
  Esamina$ = UCASE$(LEFT$(RecordAlimento.Alimento, Lunghezza))
  IF Esamina$ = elem$ THEN
    PRINT RecordAlimento.alimento; RecordAlimento.tipo;
    PRINT RecordAlimento.calorie
    trovato = 1
  ELSEIF Esamina$ > elem$ THEN
    EXIT DO
  END IF
  NumeroRecord = NumeroRecord + 1
LOOP
SEEK #1, 1
```

(Si assume che i dati sono ordinati alfabeticamente in base al nome dell'alimento. Se non fosse così, dovrebbero venire riordinati; l'ordinamento viene trattato nel prossimo capitolo).

La variabile uguale viene azzerata prima di entrare nel DO...LOOP e ad ogni record corrispondente viene posta uguale ad uno: questo valore serve come indicatore che è stato trovato almeno un record corrispondente. Il ciclo termina quando il nome dell'alimento letto dal file è superiore alla chiave e, se non ne è stato trovato nessuno corrispondente, è pari a zero e viene visualizzato un messaggio appropriato, ad esempio:

```
IF trovato = 0 THEN PRINT elem$; " non trovato."
```

```
Digitare il nome dell'alimento da cercare: ? noccioline
NOCCIOLINE non trovato
```

Terminata una ricerca viene richiesto se si intende compierne un'altra: se si risponde **s** al prompt viene richiesto un nuovo nome da ricercare; se si digita **n**, il file viene chiuso ed il sottoprogramma termina.

Il sottoprogramma RicercaRecord viene listato di seguito: lo si digiti e lo si provi con alcune chiavi di ricerca.

```
SUB RicercaRecord (nomefile$)
  CLS

  ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento

  ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)

  ' Legge la chiave di ricerca
  DO
    INPUT "Digitare il nome dell'alimento da cercare: "; elem$
    Lungh = LEN(elem$)
    elem$ = UCASE$(elem$)

    ' Visualizza i record corrispondenti del file
    NumeroRecord = 1: trovato = 0
    DO UNTIL EOF(1)
      GET #1, NumeroRecord, RecordAlimento
      Esamina$ = UCASE$(LEFT$(RecordAlimento.alimento, Lungh))
      IF Esamina$ = elem$ THEN
        PRINT RecordAlimento.alimento; RecordAlimento.tipo;
        PRINT RecordAlimento.calorie
        trovato = 1
      ELSEIF Esamina$  elem$ THEN
        EXIT DO
      END IF
```

```
    NumeroRecord = NumeroRecord + 1
  LOOP
  SEEK #1, 1
  IF trovato = 0 THEN PRINT elem$; " non trovato."
  PRINT : PRINT "Un'altra ricerca (S o N) "
  Ripeti$ = UCASE$(INPUT$(1))
  IF Ripeti$ <> "S" THEN EXIT DO
LOOP
LOCATE 25, 1: PRINT "Premere un tasto per continuare";
attesa$ = INPUT$(1)
CLOSE #1

END SUB
```

La Figura 8.8 mostra i risultati di quattro ricerche.

```
Digitare il nome dell'alimento da cercare: ? torta
Torta                meringata          105
Torta                pane degli angeli   77

Un'altra ricerca (S o N)
Digitare il nome dell'alimento da cercare: ? ca
Caciotta             fresca              30
Carne                ai ferri            93
Carote               crude               12

Un'altra ricerca (S o N)
Digitare il nome dell'alimento da cercare: ? avoc
Avocado              fresco               48

Un'altra ricerca (S o N)
Digitare il nome dell'alimento da cercare: ? p
Pane                 integrale           69
Pollo                arrosto             83

Un'altra ricerca (S o N)
Premere un tasto per continuare
```

Figura 8.8 Risultato di quattro ricerche

8.4.5 Caso della scelta errata

Tutti commettono errori alla tastiera, per cui è necessario prevenire il caso di un input che non è nel menu: questo viene trattato da un breve DO...LOOP che emette un suono per segnalare quando occorre premere un tasto ed accetta solo A, L, T, R ed U.

```
DO
  BEEP: Scelta$ = UCASE$(INPUT$(1))
LOOP WHILE INSTR("ALTRU", Scelta$) = 0
```

A questo punti sono stati realizzati il programma principale e quattro sottoprogrammi, completando il Programma 8.3, FILE VALORI NUTRIZIONALI FINALE. Si possono aggiungere altri record al file dell'alimento, leggere l'intero file, leggere record selezionati in base al numero e ricercare record aventi un determinato nome; è naturalmente possibile personalizzare il programma per adattarlo alle proprie esigenze personali.

```
DECLARE SUB AggiungiRecord (nomefile$)
DECLARE SUB LeggiRecord (nomefile$)
DECLARE SUB TrovaRecord (nomefile$)
DECLARE SUB RicercaRecord (nomefile$)
REM ** VALORI NUTRIZIONALI FINALE **
' Capitolo 8
' Microsoft QuickBASIC  4.5 File: UQB0803.BAS

REM ** Richiede il drive ed il nome del file **
CLS : DEFINT A-Z
LOCATE 5, 5: INPUT "Drive contenente il file: "; drive$
drive$ = LEFT$(drive$, 1)
LOCATE 7, 5: INPUT "Nome del file con estensione: "; nomefile$
nomefile$ = drive$ + ":" + nomefile$

REM ** Definisce la struttura del record **
TYPE TipoAlimento
  alimento AS STRING * 18
  tipo AS STRING * 15
  calorie AS INTEGER
END TYPE

REM ** Scrive il menu **
DO
  LOCATE 5, 5: PRINT SPACE$(70); : LOCATE 7, 5: PRINT SPACE$(70);
  LOCATE 3, 23: PRINT STRING$(35, 220);
  LOCATE 19, 23: PRINT STRING$(35, 223);
  FOR riga = 4 TO 18
    LOCATE riga, 23: PRINT CHR$(221);
    LOCATE riga, 57: PRINT CHR$(222);
  NEXT riga
  LOCATE 5, 29: PRINT "MENU FILE AD ACCESSO CASUALE"
```

```
  LOCATE 9, 29: PRINT "(A)ggiunge record";
  LOCATE 11, 29: PRINT "(L)egge tutti i record";
  LOCATE 13, 29: PRINT "(T)rova un record specifico";
  LOCATE 15, 29: PRINT "(R)icerca record";
  LOCATE 17, 29: PRINT "(U)scita";

  REM ** Compie la scelta **
  LOCATE 22, 3: PRINT SPACE$(70);
  LOCATE 22, 3: PRINT "Digitare un'opzione (A, L, T, R, od U)";
  DO
    BEEP: TastoScelto$ = UCASE$(INPUT$(1))
  LOOP WHILE INSTR("ALTRU", TastoScelto$) = 0
  SELECT CASE TastoScelto$
    CASE "A"
      CALL AggiungiRecord(nomefile$)
    CASE "L"
      CALL LeggiRecord(nomefile$)
    CASE "T"
      CALL TrovaRecord(nomefile$)
    CASE "R"
      CALL RicercaRecord(nomefile$)
    CASE "U"
      EXIT DO
  END SELECT
  CLS
LOOP
CLS : END

SUB AggiungiRecord (nomefile$)
  CLS
  ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento
  ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)
  ' Calcola il numero di record con LOF
  NumeroRecord = LOF(1) \ LEN(RecordAlimento)
  ' Aggiunge i record e chiude il file

  DO
    INPUT "Nome dell'alimento ?", RecordAlimento.alimento
    INPUT "Tipo dell'alimento ?", RecordAlimento.tipo
    INPUT "Calorie per oncia ?", RecordAlimento.calorie
    NumeroRecord = NumeroRecord + 1
    PUT #1, NumeroRecord, RecordAlimento
    INPUT "Altro (S o N) ? ", ancora$
    PRINT
  LOOP UNTIL UCASE$(ancora$) = "N"
  CLOSE #1
END SUB
```

```
SUB TrovaRecord (nomefile$)
  CONST Falso = 0, Vero = NOT Falso
  CLS
   ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento
  ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)
  ' Calcola il numero di record con LOF
  NumeroRecord = LOF(1) \ LEN(RecordAlimento)
  ' Trova il record richiesto
  LeggeAltri = Vero

  DO
    PRINT "Numero del record: ";
    INPUT "", NumeroRecord
    IF NumeroRecord > 0 AND NumeroRecord <= NumeroDiRecord THEN
      GET #1, NumeroRecord, RecordAlimento
      PRINT RecordAlimento.alimento; RecordAlimento.tipo;
      PRINT RecordAlimento.calorie
    ELSEIF NumeroRecord = 0 THEN
      LeggeAltri = Falso
    ELSE
      PRINT "Valore fuori dai limiti."
    END IF
    PRINT
  LOOP WHILE LeggeAltri
  LOCATE 25, 1: PRINT "Premere un tasto per continuare";
  attesa$ = INPUT$(1)
  CLOSE #1
END SUB

SUB LeggiRecord (nomefile$)
  CLS
  ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento
  ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)
  ' Calcola il numero di record con LOF
  NumeroRecord = LOF(1) \ LEN(RecordAlimento)
  ' Legge e visualizza i record
  FOR NumeroRecord = 1 TO NumeroDiRecord
    GET #1, NumeroRecord, RecordAlimento
    PRINT RecordAlimento.alimento; RoodRecord.tipo;
    PRINT RecordAlimento.calorie
  NEXT NumeroRecord
  LOCATE 25, 1: PRINT "Premere un tasto per continuare";
  attesa$ = INPUT$(1)
  CLOSE #1
END SUB
```

```
SUB RicercaRecord (nomefile$)
  CLS
  ' Dichiara il tipo di RecordAlimento
  DIM RecordAlimento AS TipoAlimento
  ' Apre il file ad accesso casuale
  OPEN nomefile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)
  ' Legge la chiave di ricerca
  DO
    INPUT "Digitare il nome dell'alimento da cercare: "; elem$
    Lungh = LEN(elem$)
    elem$ = UCASE$(elem$)
    ' Visualizza i record corrispondenti del file
    NumeroRecord = 1: trovato = 0
    DO UNTIL EOF(1)
      GET #1, NumeroRecord, RecordAlimento
      Esamina$ = UCASE$(LEFT$(RecordAlimento.alimento, Lungh))
      IF Esamina$ = elem$ THEN
        PRINT RecordAlimento.alimento; RecordAlimento.tipo;
        PRINT RecordAlimento.calorie
        trovato = 1
      ELSEIF Esamina$ > elem$ THEN
        EXIT DO
      END IF
      NumeroRecord = NumeroRecord + 1
    LOOP
    SEEK #1, 1
    IF trovato = 0 THEN PRINT elem$; " non trovato."
    PRINT : PRINT "Un'altra ricerca (S o N) "
    Ripeti$ = UCASE$(INPUT$(1))
    IF Ripeti$  "S" THEN EXIT DO
  LOOP
  LOCATE 25, 1: PRINT "Premere un tasto per continuare";
  attesa$ = INPUT$(1)
  CLOSE #1
END SUB
```

Programma 8.3 FILE VALORI NUTRIZIONALI FINALE

8.5 Sommario

Questo capitolo è stato dedicato ad i file ad accesso casuale che, aperti con al clausola FOR RANDOM, possono essere sia letti che scritti; la lunghezza specificata dalla clausola LEN dell'istruzione OPEN definisce la lunghezza dei record memorizzati nel file, mentre il blocco TYPE...END TYPE definisce i tipi delle variabili utilizzate. Questo metodo di definizione del tipo della variabile consente di riunire campi numerici e di stringhe

all'interno dello stesso record e rende automatico il calcolo della lunghezza del record. Per scrivere un record in un file ad accesso casuale si utilizza l'istruzione PUT #, mentre per leggerlo si utilizza GET #; la lunghezza del file ad accesso casuale viene restituita dalla function LOF(); l'istruzione CLOSE # viene utilizzata per chiudere i file ad accesso casuale.

Un programma di questo capitolo ha letto i propri dati da una serie di istruzioni DATA, gli altri due si sono serviti della tastiera per l'inserimento dei dati ed hanno utilizzato dei menu per selezionare l'azione desiderata da un elenco di possibili scelte, per aggiungere record al file, leggere tutti i suoi record, un record avente numero specificato ed i record del file corrispondenti ad un dato nome, completo o parziale.

Capitolo 9

Gestione dei file

9.1 Copia di un file sequenziale
9.2 Copia e modifica di un file sequenziale
9.3 Ordinamento di un file ad accesso casuale
9.4 Sommario

Col passare del tempo i dati memorizzati nei file possono divenire obsoleti, per cui le informazioni soggette a cambiamento dovrebbero essere aggiornate periodicamente: questo capitolo presenta numerosi programmi di utilità che consentono di copiare e modificare i record di un file. Alcune delle utility possono essere eseguite dal DOS ed altre dall'ambiente di QuickBASIC; tuttavia possono presentarsi delle circostanze nelle quali sarebbe preferibile copiare o modificare file all'interno di un programma scritto in QuickBASIC, ad esempio per alterare i record del file dati di un programma per adattarli ad una specifica situazione senza abbandonare l'esecuzione del programma stesso.

Si potrebbe voler cancellare alcuni record od aggiungerne altri: le utility presentate in questo capitolo possono essere utilizzate per modificare il file dei dati. La prima utility illustrata consente di copiare e visualizzare un file sequenziale; segue un'utility più complessa che consente anche l'aggiunta di nuovi record, il loro inserimento all'interno dei file e la loro cancellazione, sempre per file sequenziali; l'ultima utility di questo capitolo illustra come ordinare i record di un file ad accesso casuale dopo che siano stati mescolati casualmente.

9.1 Copia di un file sequenziale

Il programma che segue duplica i dati di un file ed è un utile strumento per gestire i file sequenziali consentendo di creare copie di sicurezza del file dati, anche da un disco ad un altro; l'MS-DOS ha un comando di copia con la sintassi seguente:

```
A>COPY nome file B:          <copia un file (nome file nel disco del drive B:)
```

Tuttavia a volte potrebbe essere necessario copiare un file in una parte di programma realizzato in QuickBASIC senza doverlo interrompere per utilizzare COPY; per copiare un file sequenziale un programma dovrebbe eseguire i seguenti passi:

- leggere il nome del vecchio file da copiare ed il drive in cui si trova;
- leggere il nome del nuovo file (la futura copia) ed il drive in cui si trova;
- utilizzare le informazioni ottenute nei passi 1 e 2 per creare le specifiche necessarie ad accedere ai due file;
- attendere che i dischi siano posti nei drive appropriati;
- aprire il file da copiare in input ed il file che riceverà la copia in output;
- leggere i record dal vecchio file e scriverli nel nuovo file;
- chiudere entrambi i file;
- emettere un messaggio che indichi che il file originale è stato copiato;
- esaminare il file per verificare che sia stato copiato correttamente.

La stesura del programma di copia dei file può iniziare da un abbozzo contenente un programma principale e dei sottoprogrammi che si limitino a scrivere sul video i propri nomi.

Il Programma 9.1, COPIA DI FILE SEQUENZIALE, chiama tre sottoprogrammi:

```
CALL NomeDrive(VecchioFile$, NuovoFile$)    <legge i drive ed i nomi dei file
CALL CopiaFile(VecchioFile$, NuovoFile$)    <copia il file
CALL EsaminaFile(FileDaEsaminare$)          <verifica che la copia sia corretta
```

Segue il listato.

```
DECLARE SUB NomeDrive (VecchioFile$, NuovoFile$)
DECLARE SUB CopiaFile (VecchioFile$, NuovoFile$)
DECLARE SUB EsaminaFile (FileDaEsaminare$)

REM ** COPIA UN FILE SEQUENZIALE **
' Capitolo 9
' Microsoft QuickBASIC 4.5  File: UQB0901.BAS

CLS : PRINT "Programma 9.1, COPIA UN FILE SEQUENZIALE"
PRINT

DIM SHARED Text$(1 TO 5)                'Usato in EsaminaFile.Sub
CALL NomeDrive(VecchioFile$, NuovoFile$)

PRINT : PRINT "Premere un tasto per continuare."
inizio$ = INPUT$(1)
CALL CopiaFile(VecchioFile$, NuovoFile$)
PRINT : PRINT "La copia del file e' terminata."
```

```
PRINT : PRINT "Premere S per esaminare il nuovo file"
PRINT : PRINT "premere un tasto diverso da S per uscire."
scelta$ = UCASE$(INPUT$(1))
IF scelta$ = "S" THEN CALL EsaminaFile(FileDaEsaminare$)
END

SUB CopiaFile (VecchioFile$, NuovoFile$)
   PRINT : PRINT "Sottoprogramma CopiaFile"
END SUB

SUB NomeDrive (VecchioFile$, NuovoFile$)
   PRINT "Sottoprogramma NomeDrive"
END SUB

SUB EsaminaFile (FileDaEsaminare$)
   PRINT : PRINT "Sottoprogramma EsaminaFile"

END SUB
```

Programma 9.1 COPIA DI FILE SEQUENZIALE

Si digiti il Programma 9.1, lo si salvi con il nome UQB0901.BAS e lo si esegua.

Il programma inizia cancellando lo schermo e scrivendo il proprio nome; l'istruzione DIM SHARED crea un vettore di stringhe che sarà utilizzato nell'ultimo sottoprogramma, EsaminaFile, quindi viene chiamato il sottoprogramma NomeDrive, che per ora si limita a scrivere il proprio nome; dopo la comparsa del messaggio il programma attende la pressione di un tasto: l'output a questo punto è:

```
Programma 9.1, COPIA DI FILE SEQUENZIALE
Sottoprogramma NomeDrive
Premere un tasto per continuare.
```

Una volta premuto un tasto, il programma chiama il sottoprogramma CopiaFile, che scrive il proprio nome, una serie di messaggi ed attende nuovamente la pressione di un tasto.

L'output è ora:

```
Programma 9.1, COPIA DI FILE SEQUENZIALE
Sottoprogramma NomeDrive
Premere un tasto per continuare.
Sottoprogramma CopiaFile
La copia del file e' terminata.
Premere S per esaminare il nuovo file
Premere un tasto diverso da S per uscire.
```

Si prema S e verrà chiamato il sottoprogramma EsaminaFile, il quale a sua volta scriverà il proprio nome, producendo la schermata finale mostrata nella Figura 9.1.

```
Programma 9.1, COPIA DI FILE SEQUENZIALE
Sottoprogramma NomeDrive
Premere un tasto per continuare.
Sottoprogramma CopiaFile
La copia del file e' terminata.
Premere S per esaminare il nuovo file
premere un tasto diverso da S per uscire.
Sottoprogramma EsaminaFIle

Premere un tasto per continuare
```

Figura 9.1 Output del Programma 9.1 prima dell'aggiunta dei sottoprogrammi

Una volta verificato il flusso del programma si aggiungano i sottoprogrammi uno alla volta, in modo da poterli controllare individualmente; alcuni sottoprogrammi verranno riutilizzati in seguito, per cui dovrebbero essere salvati separatamente, in modo da potere essere inclusi quando necessario in altri programmi.

9.1.1 Aggiunta del sottoprogramma NomeDrive

Dal menu File, si selezioni Nuovo programma e si prema **Invio**, quindi si digiti la seguente linea:

```
REM ** PRO9.2 Sottoprog NomeDrive - Usare il menu Visualizza per vedere **
```

Si richiami il comando Nuova SUBroutine del menu Modifica, servendosi dei menu completi; si digiti **nomedrive** nel riquadro e si prema **Invio**, quindi si digiti il Programma 9.2, cioè il sottoprogramma NomeDrive.

```
REM ** PRO9.2 Sottoprog NomeDrive - Usare il menu Visualizza per vedere **

SUB NomeDrive (VecchioFile$, NuovoFile$)
' Legge drive e nomi dei file
   CLS
   INPUT "Nome del vecchio file"; FileDa$
   INPUT "Drive del vecchio file"; DriveDa$

   PRINT
   INPUT "Nome del nuovo file"; FileA$
```

```
  INPUT "Drive del nuovo file"; DriveA$
  VecchioFile$ = LEFT$(DriveDa$, 1) + ":" + FileDa$
  NuovoFile$ = LEFT$(DriveA$, 1) + ":" + FileA$
END SUB
```

Programma 9.2 Sottoprogramma NomeDrive

Una volta immesso il sottoprogramma, si utilizzi Salva con nome nel menu File per salvare il file contenente il programma principale di una linea ed il sottoprogramma NomeDrive come NomeDriv.Sub: in questo modo il sottoprogramma viene salvato in un file che può essere incluso in altri programmi diversi dal Programma 9.1.

A questo punto si carichi nuovamente il Programma 9.1, si apra il menu Visualizza, si selezioni il comando SUBroutine e si prema **Invio**; si porti la barra evidenziatrice fino al sottoprogramma NomeDrive e si utilizzi il tasto **Tab** per evidenziare il pulsante CANCELLA, quindi si prema **Invio**: un riquadro di comunicazione richiederà se si intende cancellare la procedura dal modulo principale. Dato che occorre cancellare il vecchio sottoprogramma di tre linee NomeDrive, si prema **INVIO** con il pulsante OK evidenziato, cancellando così il sottoprogramma che si limitava a scrivere il proprio nome.

Si vada al programma principale e si porti il cursore al suo inizio (prima lettera della prima istruzione DECLARE): per posizionare il nuovo sottoprogramma come richiesto, si apra il menu File, si selezioni il comando Merge e si prema **Invio**, digitando quindi nell'apposito riquadro il nome del file da inserire, ovverosia:

```
B:NomeDriv.Sub
```

Si prema INVIO, si cancelli l'istruzione REM che apparirà all'inizio del programma, quindi si passi al sottoprogramma NomeDrive tramite il menu Visualizza per assicurarsi che il merge sia stato eseguito correttamente: si compiano le eventuali modifiche necessarie e si salvi questa nuova versione del Programma 9.1 come UQB0901A.

Si è pronti ora per controllare il sottoprogramma NomeDrive; si mandi in esecuzione la nuova versione del Programma 9.1: il nome del programma viene mostrato sullo schermo come nell'esecuzione precedente. In risposta ai prompt del sottoprogramma NomeDrive si diano le seguenti informazioni:

Nome del vecchio file ? Vecchio.BAS	<vecchio è il vecchio file
Drive del vecchio file ? A	<vecchio è nel drive A
Nome del nuovo file ? Nuovo.BAS	<nuovo è il nuovo file
Drive del nuovo file ? B	<nuovo sarà posto nel drive B

Premere un tasto per continuare.

Questa esecuzione di prova mostra che il sottoprogramma NomeDrive funziona correttamente: si prema un tasto e verranno mostrate le informazioni del sottoprogramma CopiaFile: si prema un tasto qualsiasi, salvo S, per uscire dal programma.

9.1.2 Aggiunta del sottoprogramma CopiaFile

Il sottoprogramma CopiaFile viene aggiunto allo stesso modo di NomeDrive: si digiti il sottoprogramma CopiaFile in un file separato, si selezioni Nuovo programma nel menu File e si digiti la seguente istruzione REM:

```
REM ** PRO9.3 Sottoprog CopiaFile - Usare il menu Visualizza per vedere **
```

Si richiami il comando Nuova SUBroutine del menu Modifica, si digiti **copiafile** e si prema INVIO, quindi si digiti il Programma 9.3, cioè il sottoprogramma CopiaFile.

```
REM ** PRO9.3 Sottoprog CopiaFile - Usare il menu Visualizza per vedere **

SUB CopiaFile (VecchioFile$, NuovoFile$)

 ' Apre due file, copia i contenuti del primo nel secondo
 ' ed infine chiude i due file
   OPEN VecchioFile$ FOR INPUT AS #1
   OPEN NuovoFile$ FOR OUTPUT AS #2

   DO UNTIL EOF(1)
     LINE INPUT #1, record$
     PRINT #2, record$
   LOOP

   CLOSE #1, #2
END SUB
```

Programma 9.3 Sottoprogramma CopiaFile

Una volta immesso il sottoprogramma, si utilizzi Salva con nome nel menu File per salvare il file con il nome CopiaFil.Sub: in questo modo il sottoprogramma viene salvato in un file che può essere incluso in altri programmi.

A questo punto si carichi nuovamente il Programma 9.1 (UQB0901.BAS) e si cancelli il vecchio sottoprogramma di tre linee nello stesso modo con il quale si è cancellato il vecchio sottoprogramma di tre linee NomeDrive; si porti ora il cursore all'inizio del programma principale, si apra il menu File, si selezioni il comando Merge e si prema **Invio**, scrivendo nell'apposito riquadro il nome del file da inserire:

```
B:CopiaFil.Sub
```

Ancora una volta, si prema **Invio**, si cancelli l'istruzione REM che apparirà all'inizio del programma, quindi si passi al sottoprogramma CopiaFile tramite il menu Visualizza per

assicurarsi che il merge sia stato eseguito correttamente: si salvi questa nuova versione del Programma 9.1 come UQB0901B (Programma 9.1B).

Si può controllare il funzionamento del sottoprogramma CopiaFile creando un file breve ed utilizzando il Programma 9.1B per copiarlo; per creare il file si selezioni Nuovo programma nel menu File e si digitino le seguenti linee:

```
REM ** PROGRAMMA PER VERIFICARE IL SOTTOPROGRAMMA COPIAFILE **
' Programma senza linee numerate
' A scopo di verifica
' Salvare come: FILEREM.BAS
' Sara' copiato in: TEMP.BAS
```

Si salvi il file di testo con il nome FileRem.Bas nello stesso dischetto contenente l'ultima versione del Programma 9.1B, quindi lo si inserisca nel drive B: e si carichi ed esegua il programma, digitando i seguenti dati come input:

```
PROGRAMMA 9.1: COPIA DI UN FILE SEQUENZIALE

Nome del vecchio file ? FILEREM.BAS      <ci si assicuri di digitare l'estensione .BAS
Drive del vecchio file ? B

Nome del nuovo file ? TEMP.BAS           <nome ed estensione
Drive del nuovo file ? B

Premere un tasto per continuare.
```

Una volta premuto un tasto FileRem.Bas sarà copiato nello stesso dischetto (nel drive B) con un nuovo nome, Temp.Bas e, una volta terminata la copia, il programma emetterà nuovi messaggi.

```
La copia del file e' terminata.
Premere S per esaminare il nuovo file
Premere un tasto diverso da S per uscire.
```

Si prema un tasto diverso da S per uscire, dal momento che il sottoprogramma EsaminaFile non è stato ancora scritto.

La copia del programma può essere esaminata caricandola con il comando Apri programma del menu File: ciò è possibile dal momento che FileRem è stato salvato come programma QuickBASIC, anche se in realtà è un file di testo e non può essere quindi eseguito.

9.1.3 Aggiunta del sottoprogramma EsaminaFile

Si selezioni Nuovo programma nel menu File e si digiti la seguente istruzione REM:

```
REM ** PRO9.4 Sottoprog EsaminaFile - Usare il menu Visualizza per vedere **
```

Si richiami il comando Nuova SUBroutine del menu Modifica, si digiti **esaminafile** e si prema **Invio**, quindi si digiti il Programma 9.4, cioè il sottoprogramma EsaminaFile.

```
REM ** PRO9.4 Sottoprog EsaminaFile - Usare il menu Visualizza per vedere **

SUB EsaminaFile (FileDaEsaminare$)
' Esamina un file dati memorizzato su disco
' Le istruzioni sono mostrate nella parte inferiore dello schermo

   ' Visualizza le istruzioni nelle linee da 21 a 25
   Text$(1) = STRING$(74, 196)
   Text$(2) = "Esamina il file "+FileDaEsaminare$ + " un record per volta."
   Text$(3) = "Porre il disco nel drive " + LEFT$(FileDaEsaminare$, 1) + "."
   Text$(4) = "Premere un tasto per iniziare: apparira' un record."
   Text$(5) = "Premere la barra spaziatrice per i successivi, od U per uscire."
   CLS : DEFINT A-Z
   FOR riga = 1 TO 5
     LOCATE riga + 20, 3: PRINT Text$(riga);
   NEXT riga

   ' Definisce le linee da 1 a 20 come finestra di visualizzazione del file
   VIEW PRINT 1 TO 20

   ' Attende la pressione di un tasto.
   LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
   untasto$ = INPUT$(1)
   CLS 2                              'Cancella solo il riquadro

   ' Apre il file da esaminare in input con il numero 1.
   OPEN FileDaEsaminare$ FOR INPUT AS #1

   ' Legge un record ad ogni pressione di tasto diverso da U
   DO UNTIL EOF (1)
     LINE INPUT #1, record$
     PRINT record$
     PRINT
     tasto$ = INPUT$(1)
     IF UCASE$(tasto$) = "U" THEN EXIT DO
   LOOP

   PRINT : PRINT "Fine del file. Premere un tasto per continuare.";
   untasto$ = INPUT$(1)
   ' Chiude il file e termina
```

```
   VIEW PRINT 1 TO 25: CLS
   CLOSE #1
END SUB
```

Programma 9.4 Sottoprogramma EsaminaFile

Si cancelli il vecchio sottoprogramma di tre linee EsaminaFile dal Programma 9.1 richiamando il comando SUBroutine dal menu Visualizza, evidenziando il sottoprogramma EsaminaFile nel riquadro, utilizzando il tasto **Tab** per evidenziare il pulsante CANCELLA e premendo **Invio**. Quindi si compia il merge del nuovo sottoprogramma EsaminaFile (EsamFile.Sub) per mezzo del comando Merge del menu File e si esamini il sottoprogramma EsaminaFile nella finestra di visualizzazione per assicurarsi della riuscita del merge. Si salvi il programma rivisto insieme a tutti i sottoprogrammi con il nome UQB0905.BAS e si cambi il suo nome in Programma 9.5, COPIA ED ESAMINA UN FILE SEQUENZIALE. A questo punto si può controllare il funzionamento del sottoprogramma EsaminaFile utilizzando il Programma 9.5 (UQB0905.BAS) per copiare se stesso in un file chiamato Temp.Bas e per esaminare subito dopo la copia del programma; tra programma ed utente si svolgerà il seguente dialogo:

```
Nome del vecchio file ? UQB0905.BAS

Drive del vecchio file ? B

Nome del nuovo file ? TEMP.BAS

Drive del nuovo file ? B

Premere un tasto per continuare.

La copia del file e' terminata.

Premere S per esaminare il nuovo file

premere un tasto diverso da S per uscire.
```

Si prema **S** per esaminare il file Temp.Bas: la Figura 9.2 mostra un momento della verifica dei contenuti della copia del file, che può essere letto interamente oppure in parte, premendo il tasto **U** una volta che si sia convinti che la copia è riuscita correttamente.

```
DECLARE SUB NomeDrive (VecchioFile$, NuovoFile$)
DECLARE SUB CopiaFile (VecchioFile$, NuovoFile$)
DECLARE SUB EsaminaFile (FileDaEsaminare$)
REM ** COPIA UN FILE SEQUENZIALE **
' Capitolo 9
 Microsoft QuickBASIC 4.5  File: UQB0905.BAS

  CLS : PRINT "Programma 9.5, COPIA ED ESAMINA UN FILE SEQUENZIALE"
PRINT
DIM SHARED Text$(1 TO 5)                'Utilizzato in EsaminaFile.Sub

CALL NomeDrive(VecchioFile$, NuovoFile$)
```

```
PRINT : PRINT "Premere un tasto per continuare."
inizio$ = INPUT$(1)

CALL CopiaFile(VecchioFile$, NuovoFile$)

PRINT : PRINT "Copia del file terminata."
PRINT : PRINT "Premere S per esaminare il nuovo file"
PRINT : PRINT "premere un tasto diverso da S per uscire."

scelta$ = UCASE$(INPUT$(1))
IF scelta$ = "S" THEN CALL EsaminaFile(FileDaEsaminare$)
END

SUB CopiaFile (VecchioFile$, NuovoFile$)
 ' Apre due file, copia i contenuti del primo nel secondo
 ' ed infine chiude i due file
   OPEN VecchioFile$ FOR INPUT AS #1
   OPEN NuovoFile$ FOR OUTPUT AS #2

   DO UNTIL EOF(1)
     LINE INPUT #1, record$
     PRINT #2, record$
   LOOP
   CLOSE #1, #2
END SUB

SUB NomeDrive (VecchioFile$, NuovoFile$)
' Legge drive e nomi dei file

   CLS
   INPUT "Nome del vecchio file"; FileDa$
   INPUT "Drive del vecchio file"; DriveDa$
   PRINT
   INPUT "Nome del nuovo file"; FileA$
   INPUT "Drive del nuovo file"; DriveA$
   VecchioFile$ = LEFT$(DriveDa$, 1) + ":" + FileDa$
   NuovoFile$ = LEFT$(DriveA$, 1) + ":" + FileA$
END SUB

SUB EsaminaFile (FileDaEsaminare$)
' Esamina un file dati memorizzato su disco
' Le istruzioni sono mostrate nella parte inferiore dello schermo

   ' Visualizza le istruzioni nelle linee da 21 a 25
   Text$(1) = STRING$(74, 196)
   Text$(2) = "Esamina il file "+FileDaEsaminare$ + " un record per volta."
   Text$(3) = "Porre il disco nel drive " + LEFT$(FileDaEsaminare$, 1) + "."
   Text$(4) = "Premere un tasto per iniziare: apparira' un record."
   Text$(5) = "Premere la barra spaziatrice per i successivi, od U per uscire."
```

```
  CLS : DEFINT A-Z
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT Text$(riga);
  NEXT riga

  ' Definisce le linee da 1 a 20 come finestra di visualizzazione del file
  VIEW PRINT 1 TO 20
  ' Attende la pressione di un tasto.
  LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
  untasto$ = INPUT$(1)
  CLS 2                              'Cancella solo il riquadro
  ' Apre il file da esaminare in input con il numero 1
  OPEN FileDaEsaminare$ FOR INPUT AS #1

  ' Legge un record ad ogni pressione di tasto diverso da U
  DO UNTIL EOF(1)
    LINE INPUT #1, record$
    PRINT record$
    PRINT
    tasto$ = INPUT$(1)
    IF UCASE$(tasto$) = "U" THEN EXIT DO
  LOOP

  PRINT : PRINT "Fine del file. Premere un tasto per continuare.";
  untasto$ = INPUT$(1)
  ' Chiude il file e termina
  VIEW PRINT 1 TO 25: CLS
  CLOSE #1
END SUB
```

Programma 9.5 COPIA ED ESAMINA UN FILE

Il sottoprogramma EsaminaFile legge ciascun record con l'istruzione LINE INPUT # e separa con una linea vuota le linee del programma per facilitarne la lettura; nel caso si intendesse modificare i meccanismi di lettura e scrittura sullo schermo dei record si possono scrivere sottoprogrammi personalizzati per presentare i dati come si desidera.

```
DECLARE SUB NomeDrive (VecchioFile$, NuovoFile$)
DECLARE SUB CopiaFile (VecchioFile$, NuovoFile$)
DECLARE SUB EsaminaFile   (FileDaEsaminare$)
REM ** COPIA UN FILE SEQUENZIALE **
' Capitolo 9
' Microsoft QuickBASIC 4.5  File: UQB0905.BAS
CLS: PRINT  "Programma 9.5, COPIA UN FILE SEQUENZIALE"
PRINT
```

```
Esamina il file B:TEMP.BAS un record per volta.

Porre il disco nel drive B.

Premere un tasto per iniziare: apparira' un record.

Premere la barra spaziatrice per i successivi od U per uscire.
```

Figura 9.2 Esame della copia del file

Viene ora presentato un breve programma che crea un file nel quale i record sono le lettere dell'alfabeto, CREA IL FILE DELLE LETTERE DALLA ALLA A Z: si digiti ed esegua tale programma, quindi si utilizzi il programma 9.5 per copiare il file da esso prodotto (Alfabeto.Dat) in un file temporaneo (Temp.Dat). Alfabeto.Dat sarà riutilizzato in seguito; la Figura 9.3 mostra l'esame del file Temp.Dat

```
REM ** CREA IL FILE DELLE LETTERE DA A A Z **
' Capitolo 9
' Microsoft QuickBASIC 4.5  File: UQB0906.BAS

' Inizializza le variabili ed apre il file
DEFINT A-Z
OPEN "B:Alfabeto.Dat" FOR OUTPUT AS #1

' Crea i record contenenti le lettere
FOR k = 1 TO 26
   PRINT #1, CHR$(k + 64)
NEXT k

' Chiude il file e termina
CLOSE #1
END
```

Programma 9.6 CREA IL FILE DELLE LETTERE DALLA A ALLA Z

```
A

B

C

D

E

F

G

H

I
```

```
Esamina il file B:TEMP.BAS un record per volta.
Porre il disco nel drive B.
Premere un tasto per iniziare: apparira' un record.
Premere la barra spaziatrice per i successivi od U per uscire.
```

Figura 9.3 Esame del file Temp.Dat

9.2 Copia e modifica di un file sequenziale

Un file sequenziale è stato creato, copiato ed esaminato nella parte precedente di questo capitolo: per poter modificare i dati di questo file è necessario un programma di utilità a tale scopo.

Il Programma 9.7, COPIA E MODIFICA UN PROGRAMMA SEQUENZIALE è un abbozzo di programma che contiene lo spazio per i sottoprogrammi per aggiungere, inserire, cancellare ed esaminare i dati di un file; i sottoprogrammi saranno aggiunti uno per volta.

Si digiti il Programma 9.7 e lo si mandi in esecuzione per vederne il flusso; la schermata generata è simile a quella del Programma 9.1: viene visualizzato il nome del programma, quindi viene chiamato il sottoprogramma NomeDrive, che al momento si limita a scrivere il proprio nome. Il programma attende dopo avere eseguito il sottoprogramma NomeDrive; premendo un tasto dopo la comparsa del messaggio "Premere un tasto per continuare" viene chiamato il sottoprogramma CopiaModifica, dopo l'esecuzione del quale vengono visualizzati i messaggi che seguono:

```
PRINT "Copia e modifica del file terminate."

PRINT

PRINT "Aggiungere ulteriori record a  "; NuovoFile$; " (S o N)"
Un DO...LOOP emette un suono ed attende la risposta dell'utente:

DO
  BEEP: CosaFare$ = UCASE$(INPUT$(1))
LOOP WHILE INSTR("SN", CosaFare$) = 0
```

Se si risponde premendo S viene chiamato il sottoprogramma AggiungiFile, che scrive il proprio nome e ritorna al programma principale, il quale a sua volta visualizza ulteriori messaggi; il dialogo tra programma ed utente è il seguente:

```
PROGRAMMA 9.7, COPIA E MODIFICA UN FILE SEQUENZIALE
Sottoprogramma NomeDrive
Premere un tasto per continuare.
Sottoprogramma CopiaModifica
Copia e modifica del file terminate.
Aggiungere ulteriori record a  (S o N)?
Sottoprogramma AggiungiFile
Premere S per esaminare il nuovo file
Premere un tasto diverso da S per uscire
```

Se si risponde premendo N il sottoprogramma AggiungiFile non viene eseguito ed in questo caso il dialogo è:

```
PROGRAMMA 9.7, COPIA E MODIFICA UN FILE SEQUENZIALE
Sottoprogramma NomeDrive
Premere un tasto per continuare.
Sottoprogramma CopiaModifica
Copia e modifica del file terminate.
Aggiungere ulteriori record a  (S o N)?
Premere S per esaminare un nuovo file
Premere un tasto diverso da S per uscire
```

Viene quindi cancellato lo schermo e richiesto all'utente se desidera esaminare il file: si risponda premendo S in caso affermativo, oppure si esca dal programma. A questo punto si possono aggiungere i sottoprogrammi sostituendo quelli temporanei dell'abbozzo.

9.2.1 Aggiunta dei sottoprogrammi NuovoDrive ed EsaminaFile

Si sostituisca al sottoprogramma temporaneo NomeDrive il sottoprogramma corrispondente del Programma UQB0905, che consente l'introduzione da tastiera del drive e del nome

del file da utilizzare; analogamente si sostituisca al sottoprogramma temporaneo EsaminaFile la controparte del programma UQB0905, per poter esaminare i contenuti del file una volta completate le modifiche eseguite dai sottoprogrammi CopiaModifica e AggiungiFile.

9.2.2 Aggiunta del sottoprogramma CopiaModifica

Dal momento che il sottoprogramma CopiaModifica non è stato ancora utilizzato, si digiti il Programma 9.8, esattamente come gli altri sottoprogrammi di questo capitolo, quindi si cancelli il sottoprogramma di tre linee utilizzato per visualizzare il nome del sottoprogramma e si compia il merge nel Programma 9.7.

```
REM ** PRO9.8 Sottoprog CopiaModifica - Usare Visualizza per vedere **
SUB CopiaModifica (VecchioFile$, NuovoFile$)
' Consente di leggere un record dal file vecchio
' copiare un record nel file nuovo
' od inserire un nuovo record nel file vecchio
   ' Inizializza, mostra il menu ed apre la finestra di visualizzazione
   CLS : DEFINT A-Z
   LOCATE 19, 3: PRINT STRING$(74, 196)
   LOCATE 20, 3: PRINT "Scegliere un'opzione (L, C, I, od U):"
   LOCATE 21, 3: PRINT "  (L)egge un record da "; VecchioFile$
   LOCATE 22, 3: PRINT "  (C)opia il record visualizzato in "; NuovoFile$
   LOCATE 23, 3: PRINT "  (I)nserisce un nuovo record prima del corrente"
   LOCATE 24, 3: PRINT "  (U)scita";
   VIEW PRINT 1 TO 18
   ' Apre i file ed esegue l'opzione selezionata
   OPEN VecchioFile$ FOR INPUT AS #1
   OPEN NuovoFile$ FOR OUTPUT AS #2
   DO
     DO
       BEEP: CosaFare$ = UCASE$(INPUT$(1))
       LOCATE 18, 1: PRINT STRING$(79, 32);
     LOOP WHILE INSTR("LCIU", CosaFare$) = 0
     SELECT CASE CosaFare$
       CASE "L"
         IF NOT EOF(1) THEN
           LINE INPUT #1, VecchioRecord$
           CLS 2: PRINT VecchioRecord$
         ELSE
           LOCATE 18, 1: PRINT "Sono stati letti tutti i record.";
         END IF
       CASE "C"
         PRINT #2, VecchioRecord$
         LOCATE 18, 1: PRINT "Il record corrente e' copiato in "; NuovoFile$;
       CASE "I"
         LOCATE 10, 1: PRINT "Digitare il nuovo record e premere INVIO:"
```

```
        LINE INPUT NuovoRecord$
        PRINT #2, NuovoRecord$
        CLS 2: PRINT VecchioRecord$
      CASE "U"
        EXIT DO
    END SELECT
  LOOP
  ' Chiude i file e termina il sottoprogramma
  CLOSE #1, #2
END SUB
```

Programma 9.8 Sottoprogramma CopiaModifica

Il sottoprogramma CopiaModifica consente di eseguire una delle quattro opzioni seguenti:

- leggere un record dal file vecchio;
- copiare il record visualizzato sullo schermo (corrente) nel nuovo file;
- inserire un nuovo record prima di quello corrente;
- uscire dal sottoprogramma.

Alla chiamata del sottoprogramma CopiaModifica, questo visualizza nella parte inferiore dello schermo le seguenti istruzioni ed apre due file, uno in input e l'altro in output.

```
Scegliere un'opzione (L, C, I, od U):
  (L)egge un record da VecchioFile$
  (C)opia il record visualizzato in NuovoFile$
  (I)nserisce un nuovo record prima del corrente
  (U)scita
```

Il nome del file da modificare e del nuovo file contenente il risultato delle modifiche sono sostituiti al posto di VecchioFile$ e NuovoFile$ nell'elenco delle istruzioni; una volta premuto **L**, **C**, **I** od **U** viene eseguita la porzione appropriata della SELECT CASE.

9.2.3 Aggiunta del Sottoprogramma AggiungiFile

L'ultimo sottoprogramma da aggiungere è il Programma 9.9, cioè il Sottoprogramma AggiungiFile: si cancelli il sottoprogramma temporaneo di tre linee corrispondente e si compia il merge del Programma 9.9 al suo posto, come è stato fatto per glia altri sottoprogrammi.

```
REM ** PRO9.9 Sottoprog AggiungiFile - Usare Visualizza per vedere **
SUB AggiungiFile (NomeFile$)
```

```
' Aggiunge informazioni a NomeFile$
' che comprende il drive a dischetti designato.

  ' Inizializza e scrive le istruzioni nelle linee da 21 a 25
  CLS : DEFINT A-Z
  text$(1) = STRING$(74, 196)
  text$(2) = "Aggiunge record a" + NomeFile$
  text$(3) = "Premere INVIO per iniziare"
  text$(4) = "Al prompt (>), digitare un record e premere INVIO."
  text$(5) = "Premere INVIO senza digitare alcun dato per finire."
  FOR riga = 1 TO 5
    LOCATE riga + 20, 3: PRINT text$(riga);
  NEXT riga

  ' Definisce le linee da 1 a 20 come finestra di visualizzazione,
  ' quindi attende la pressione di un tasto per iniziare.
  VIEW PRINT 1 TO 20
  LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
  untasto$ = INPUT$(1)
  CLS 2                        'Cancella solo la finestra di visualizzazione

  ' Apre il file in cui aggiungere i dati come #1
  ' quindi legge i record dalla tastiera e li aggiunge al file.
  OPEN NomeFile$ FOR APPEND AS #1
  DO
    LINE INPUT "> "; record$
    PRINT
    IF record$ = "" THEN EXIT DO
    PRINT #1, record$
  LOOP
  ' Chiude il file e termina il sottoprogramma.
  CLOSE #1
END SUB
```

Programma 9.9 Sottoprogramma AggiungiFile

Le istruzioni di AggiungiFile sono visualizzate nella parte inferiore dello schermo, mentre i prompt ed i record immessi da tastiera, che sono aggiunti alla fine del file, compaiono in un riquadro nella parte alta. Con questo sottoprogramma viene completato il programma di copia e modifica dei file: lo si salvi con il nome UQB0910.BAS

9.2.4 Utilizzo del programma COPIA E MODIFICA UN FILE SEQUENZIALE

Il Programma 9.6 ha creato un file di dati, Alfabeto.Dat, che sarà utilizzato ora per verificare il Programma 9.10, COPIA E MODIFICA UN FILE SEQUENZIALE. La Figura 9.4 mostra

il dialogo di apertura, che richiede i nomi dei file e dei drive a dischetti da utilizzare: al prompt "Premere un tasto per continuare" si inserisca nel drive B il dischetto contenente il file Alfabeto.Dat.

```
Nome del vecchio file? ALFABETO.DAT
Drive del vecchio file? B
Nome del nuovo file? ALFAMODI.DAT
Drive del nuovo file? B
Premere un tasto per continuare
```

Figura 9.4 Dialogo di apertura del Programma 9.6

Una volta premuto un tasto viene chiamato il sottoprogramma CopiaModifica e le istruzioni vengono visualizzate nella parte inferiore dello schermo.

```
Scegliere un'opzione (L, C, I, od U):
  (L)egge un record da VecchioFile$
  (C)opia il record visualizzato in NuovoFile$
  (I)nserisce un nuovo record prima del corrente
  (U)scita
```

A questo punto si eseguano i passi elencati di seguito:

- si prema L per leggere un record: il primo record di Alfabeto.Dat, A, apparirà nella finestra di visualizzazione;
- non si copi la lettera A: si prema invece nuovamente L per leggere un altro record e la lettera B apparirà nella finestra di visualizzazione;
- si prema C per copiare la lettera B nel file AlfaModi.Dat: B verrà copiata, ma resterà anche sullo schermo, nella cui parte inferiore, appena sopra le istruzioni, comparirà il messaggio mostrato nella Figura 9.5;

```
B


Il record corrente e' copiato in B:ALFAMODI.DAT


Scegliere un'opzione (L, C, I, od U):
  (L)egge un record da VecchioFile$
```

```
(C)opia il record visualizzato in NuovoFile$
(I)nserisce un nuovo record prima del corrente
(U)scita
```

Figura 9.5 Conferma della copia di un record

- si prema L per leggere il record successivo di Alfabeto.Dat: apparirà la lettera C, che non dovrà essere copiata;
- si prosegua lungo l'alfabeto, sempre allo stesso modo, leggendo tutti i record da A a Z e copiando solo B, D ed F nel file AlfaModi.Dat.

```
DECLARE SUB NomeDrive (VecchioFile$, NuovoFile$)
DECLARE SUB CopiaModifica (VecchioFile$, NuovoFile$)
DECLARE SUB EsaminaFile (FileDaEsaminare$)
DECLARE SUB AggiungiFile (NomeFile$)
REM ** COPIA E MODIFICA UN FILE SEQUENZIALE **
' Capitolo 9
' Microsoft QuickBASIC 4.5  File: UQB0910.BAS
' Inizializza
DEFINT A-Z: CLS
PRINT "Programma 9.10, COPIA E MODIFICA UN FILE SEQUENZIALE"
PRINT

' Chiama le SUBroutine una alla volta
CALL NomeDrive(VecchioFile$, NuovoFile$)
PRINT : PRINT "Premere un tasto per continuare."
inizio$ = INPUT$(1)

CALL CopiaModifica(VecchioFile$, NuovoFile$)
PRINT : PRINT "Copia e modifica del file terminate."

' Vede se aggiungere record al file
PRINT
PRINT "Aggiunge ulteriori record a"; NuovoFile$; " (S or N)?"
DO
   BEEP: CosaFare$ = UCASE$(INPUT$(1))
LOOP WHILE INSTR("SN", CosaFare$) = 0

' Se si desiderano altri record chiama il sottoprogramma
IF CosaFare$ = "S" THEN CALL AggiungiFile(NuovoFile$)
' Vede cosa fare dopo
PRINT "Premere S per esaminare il nuovo file"
PRINT : PRINT "Premere un tasto diverso da S per uscire"
scelta$ = UCASE$(INPUT$(1))

' Se desidera esaminare il file chiama il sottoprogramma
IF scelta$ = "S" THEN CALL EsaminaFile(FileDaEsaminare$)
END
```

```
SUB AggiungiFile (NomeFile$)
' Aggiunge informazioni a NomeFile$
' che comprende il drive a dischetti designato.

   ' Inizializza e scrive le istruzioni nelle linee da 21 a 25
   CLS : DEFINT A-Z
   text$(1) = STRING$(74, 196)
   text$(2) = "Aggiunge record a" + NomeFile$
   text$(3) = "Premere INVIO per iniziare"
   text$(4) = "Al prompt (>), digitare un record e premere INVIO."
   text$(5) = "Premere INVIO senza digitare alcun dato per finire."
   FOR riga = 1 TO 5
     LOCATE riga + 20, 3: PRINT text$(riga);
   NEXT riga

   ' Definisce le linee da 1 a 20 come finestra di visualizzazione,
   ' quindi attende la pressione di un tasto per iniziare
   VIEW PRINT 1 TO 20
   LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
   untasto$ = INPUT$(1)
   CLS 2                        'Cancella solo la finestra di visualizzazione

   ' Apre il file in cui aggiungere i dati come #1
   ' quindi legge i record dalla tastiera e li aggiunge al file
   OPEN NomeFile$ FOR APPEND AS #1
   DO
     LINE INPUT "> "; record$
     PRINT
     IF record$ = "" THEN EXIT DO
     PRINT #1, record$
   LOOP

   ' Chiude il file e termina il sottoprogramma
   CLOSE #1
END SUB

SUB CopiaModifica (VecchioFile$, NuovoFile$)
' Consente di leggere un record dal file vecchio
' copiare un record nel file nuovo
' od inserire un nuovo record nel file vecchio

   ' Inizializza, mostra il menu ed apre la finestra di visualizzazione
   CLS : DEFINT A-Z
   LOCATE 19, 3: PRINT STRING$(74, 196)
   LOCATE 20, 3: PRINT "Scegliere un'opzione (L, C, I, od U):"
   LOCATE 21, 3: PRINT "  (L)egge un record da "; VecchioFile$
   LOCATE 22, 3: PRINT "  (C)opia il record visualizzato in "; NuovoFile$
   LOCATE 23, 3: PRINT "  (I)nserisce un nuovo record prima del corrente"
   LOCATE 24, 3: PRINT "  (U)scita";
```

```
  VIEW PRINT 1 TO 18

  ' Apre i file ed esegue l'opzione selezionata
  OPEN VecchioFile$ FOR INPUT AS #1
  OPEN NuovoFile$ FOR OUTPUT AS #2

  DO

    DO
      BEEP: CosaFare$ = UCASE$(INPUT$(1))
      LOCATE 18, 1: PRINT STRING$(79, 32);
    LOOP WHILE INSTR("LCIU", CosaFare$) = 0
    SELECT CASE CosaFare$
      CASE "L"
        IF NOT EOF(1) THEN
          LINE INPUT #1, VecchioRecord$
          CLS 2: PRINT VecchioRecord$
        ELSE
          LOCATE 18, 1: PRINT "Sono stati letti tutti i record.";
        END IF
      CASE "C"
        PRINT #2, VecchioRecord$
        LOCATE 18, 1: PRINT "Il record corrente e' copiato in "; NuovoFile$;
      CASE "I"
        LOCATE 10, 1: PRINT "Digitare il nuovo record e premere INVIO:"
        LINE INPUT NuovoRecord$
        PRINT #2, NuovoRecord$
        CLS 2: PRINT VecchioRecord$
      CASE "U"
        EXIT DO
    END SELECT
  LOOP

  ' Chiude i file e termina il sottoprogramma
  CLOSE #1, #2
END SUB

SUB NomeDrive (VecchioFile$, NuovoFile$)
 ' Legge il drive ed i nomi dei file

  CLS
  INPUT "Nome del vecchio file"; FileDa$
  INPUT "Drive del vecchio file"; DriveDa$
  PRINT
  INPUT "Nome del nuovo file"; FileA$
  INPUT "Drive del nuovo file"; DriveA$

  VecchioFile$ = LEFT$(DriveDa$, 1) + ":" + FileDa$
  NuovoFile$ = LEFT$(DriveA$, 1) + ":" + FileA$
```

```
END SUB
SUB EsaminaFile (FileDaEsaminare$)
' Esamina un file dati salvato su disco
' Le istruzioni sono visualizzate nella parte inferiore dello schermo

   ' Visualizza le istruzioni nelle linee da 1 a 25
   Text$(1) = STRING$(74, 196)
   Text$(2) = "Esamina il file " + FileDaEsaminare$ + " un record per volta."
   Text$(3) = "Porre il disco nel drive " + LEFT$(FileDaEsaminare$, 1) + "."
   Text$(4) = "Premere un tasto per iniziare: apparira' un record."
   Text$(5) = "Premere la barra spaziatrice per i seguenti od U per uscire."
   CLS : DEFINT A-Z

   FOR riga 1 TO 5
     LOCATE riga + 20, 3: PRINT Text$(riga);
   NEXT riga

   ' Definisce le linee da 1 a 20 come finestra di visualizzazione del file
   VIEW PRINT 1 TO 20

   ' Attende la pressione di un tasto.
   LOCATE 10, 28: PRINT "Premere un tasto per iniziare."
   untasto$ = INPUT$(1)
   CLS 2                              'Cancella solo il riquadro

   ' Apre il file da esaminare in input con il numero 1
   OPEN FileDaEsaminare FOR INPUT AS #1

    ' Legge un record ad ogni pressione di tasto diverso da U

   DO UNTIL
     LINE INPUT #1, record$
     PRINT record$
     PRINT
     tasto$ = INPUT$(1)
     IF UCASE$(tasto$) = "U" THEN EXIT DO
   LOOP

   PRINT : PRINT "Fine del file. Premere un tasto per continuare.";
   untasto$ = INPUT$(1)

   ' Chiude il file e termina
   VIEW PRINT 1 TO 25: CLS
   CLOSE #1
END SUB
```

Programma 9.10 COPIA E MODIFICA UN FILE SEQUENZIALE

Tutti i tentativi di leggere un record, una volta che sia stato letto l'ultimo, faranno comparire il seguente messaggio:

```
Sono stati letti tutti i record.
```

Si prema U per uscire: verrà visualizzato il seguente messaggio, accompagnato dall'emissione di una nota:

```
Copia e modifica del file terminate.
Aggiungere ulteriori record a alfamodi.dat (S o N)?
```

Non si aggiungeranno altri record al momento, per cui si prema N seguito da **Invio**, in risposta al quale si riceverà un altro messaggio di due linee, come segue:

```
Per esaminare il nuovo file, premere S
Premere un tasto diverso da S per uscire
```

Si prema S per esaminare i contenuti del file AlfaModi.Dat: si prema un tasto qualsiasi e verrà visualizzato il primo record, B; premendo la **Barra Spaziatrice** apparirà il secondo record, D; premendo nuovamente la **Barra Spaziatrice** apparirà la lettera F. Premendo ancora una volta la **Barra Spaziatrice** verrà visualizzato il seguente messaggio:

```
Fine del file. Premere un tasto per continuare.
```

Si prema un tasto qualsiasi per uscire dal sottoprogramma EsaminaFile ed un altro tasto per terminare il programma; in questo esempio è stato appena creato un nuovo file, AlfaModi.Dat, che contiene tre record, B, D ed F, copiati dal file Alfabeto.Dat. Si mandi nuovamente in esecuzione il Programma 9.10 e si utilizzi questa volta AlfaModi.Dat come vecchio file e Modi2.Dat come nuovo file; alla chiamata del sottoprogramma CopiaModifica, si legga il primo record di AlfaModi.Dat: nella finestra di visualizzazione apparirà la lettera B.

Si prema I per inserire un record prima della lettera B; si prema **Invio** ed il caso I sarà scelto dall'istruzione SELECT CASE, visualizzando sullo schermo il messaggio "Digitare il nuovo record e premere Invio". Si digiti A seguito da **Invio**, inserendo così il record A nel file Modi2.dat prima del record contenente la lettera B, che rimane visibile sullo schermo a disposizione delle varie opzioni del menu. Si prema C per copiare B in Modi2.dat, che ora conterrà i due record:

- A = record 1
- B = record 2

Con lo stesso procedimento si legga il record successivo di AlfaModi.Dat (D) e si inserisca la lettera C in Modi2.Dat, quindi si copi D in quel file; si ripeta la procedura per il record

successivo di AlfaModi.Dat (F), inserendo la lettera E e copiando quindi la lettera F in Modi2.Dat. A questo punto, Modi2.Dat contiene 6 record:

- 1 A
- 2 B
- 3 C
- 4 D
- 5 E
- 6 F

Se si tentasse di leggere un altro record di AlfaModi.Dat, verrebbe visualizzato il seguente messaggio:

```
Copia e modifica del file terminate.
Aggiungere ulteriori record a modi2.dat (S o N) ?
```

Si prema ora S, chiamando il sottoprogramma AggiungiFile; in risposta al messaggio "Premere un tasto per iniziare" si prema un tasto qualsiasi ed al prompt (>), si digiti la lettera dell'alfabeto successiva (G) e si prema **Invio**: G sarà aggiunta in un record alla fine del file Modi2.Dat. Si ripeta il procedimento, aggiungendo altri record a piacere, quindi si prema **Invio** senza digitare alcun dato per uscire dal DO...LOOP del sottoprogramma AggiungiFile: il file verrà chiuso ed il sottoprogramma terminerà. Il programma principale visualizzerà il messaggio:

```
Per esaminare il nuovo file, premere S
Premere un tasto diverso da S per uscire
```

Premendo S si può esaminare il nuovo file (Modi2.Dat) e premendo un tasto diverso si uscirà dal programma.

9.3 Ordinamento di un file ad accesso casuale

I record dei file ad accesso casuale possono essere cercati, recuperati ed ordinati molto più velocemente di quelli dei file sequenziali. Il Programma 9.11, MESCOLA ED ORDINA UN FILE AD ACCESSO CASUALE, è un abbozzo di programma comprendente quattro

sottoprogrammi, i quali inizialmente si limitano a visualizzare il proprio nome ed a ritornare al programma principale; questo programma è stato scritto per utilizzare il file dei cibi prodotto dal Programma 8. 2, ma si possono sostituire ai sottoprogrammi altri adatti ad altri file dati, ricordandosi anche di modificare le definizioni TYPE...END TYPE del programma principale per adattarle al tipo del file dati. Si digiti ed esegua l'abbozzo, che chiama i sottoprogrammi nel seguente ordine:

```
Programma che mescola ed ordina un file ad accesso casuale
sottoprogramma NomeFile
sottoprogramma MescolaFile
sottoprogramma LeggiRecord
sottoprogramma OrdinaFile
sottoprogramma LeggiRecord
```

Come si può vedere, innanzitutto viene dato un nome al file, quindi i record sono mescolati e vengono mostrati in ordine casuale dal sottoprogramma LeggiRecord; successivamente i file vengono ordinati alfabeticamente in base al nome dell'alimento, che si trova nel primo campo di ciascun record; il sottoprogramma LeggiRecord viene chiamato nuovamente per visualizzare i record riordinati.

```
DECLARE SUB LeggiRecord (NumeroDiRecord%)
DECLARE SUB NominaFile (NomeFile$)
DECLARE SUB MescolaFile (NomeFile$, NumeroDiRecord%)
DECLARE SUB OrdinaFile NumeroDiRecord

REM ** MESCOLA E RIORDINA UN FILE AD ACCESSO CASUALE **
' Capitolo 9
' Microsoft QuickBASIC 4.5  File: UQB0911.BAS
' Inizializza
CLS : DEFINT A-Z
PRINT "Programma che MESCOLA E RIORDINA UN FILE AD ACCESSO CASUALE"

REM ** DEFINISCE LA STRUTTURA DEL RECORD **
TYPE TipoAlimento
   alimento AS STRING * 18
   tipo AS STRING * 15
   calorie AS INTEGER
END TYPE

' Chiama i sottoprogrammi
CALL NominaFile(NomeFile$)
CALL MescolaFile(NomeFile$, NumeroDiRecord%)
CALL LeggiRecord(NumeroDiRecord%)
CALL OrdinaFile(NumeroDiRecord%)

CALL LeggiRecord(NumeroDiRecord%)

' Chiude il file e termina
CLOSE #1
```

```
END

SUB NominaFile (NomeFile$)
' Assegna un nome al file
   PRINT : PRINT "Sottoprogramma NominaFile"
END SUB

SUB LeggiRecord (NumeroDiRecord%)
' Legge i record
   PRINT : PRINT "Sottoprogramma LeggiRecord"
END SUB

SUB MescolaFile (NomeFile$, NumeroDiRecord%)
' Mescola i record
   PRINT : PRINT "Sottoprogramma MescolaFile"
END SUB

SUB OrdinaFile (NumeroDiRecord%)
' Ordina i record alfabeticamente
   PRINT : PRINT "Sottoprogramma OrdinaFile"
END SUB
```

Programma 9.11 Mescola ed ordina un file ad accesso casuale

```
REM ** PRO9.2 Sottoprogramma NominaFile - Usare Visualizza per vedere **

SUB NominaFile (NomeFile$)
' Assegna un nome al file
   PRINT
   INPUT "Nome del file con estensione"; NomeFile$
   INPUT "Drive contenente il disco"; Drive$
   PRINT : PRINT "Porre il disco di "; NomeFile$; " nel drive "; Drive$
   NomeFile$ = LEFT$(Drive$, 1) + ":" + NomeFile$
   PRINT "Premere un tasto per continuare."
   untasto$ = INPUT$(1)
END SUB
```

Programma 9.12 Sottoprogramma NominaFile

9.3.1 Aggiunta del sottoprogramma NomeFile

Il sottoprogramma NominaFile è simile al sottoprogramma NomeDrive utilizzato dal Programma 9.5 ed in questo programma ha un nome diverso per distinguerlo da quello: si consiglia di salvare separatamente i due sottoprogrammi per aiutare ad evitare confusioni durante le operazioni di merge dei sottoprogrammi nei programmi principali: si sostituisca il Programma 9.12 (sottoprogramma NomeFile) al sottoprogramma NominaFile tempora-

neo del Programma 9.11. Il nuovo sottoprogramma richiede all'utente il nome del file e del drive a dischetti da utilizzare e l'inserimento del dischetto appropriato nel drive, attendendo la pressione di un tasto prima di restituire il controllo al programma principale. Prima di proseguire si verifichi il corretto funzionamento di questa parte di programma.

9.3.2 Aggiunta del sottoprogramma MescolaFile

Il sottoprogramma MescolaFile dichiara RecordAlimento e TempAlimento come oggetti di tipo TipoAlimento per mezzo della seguente istruzione DIM:

```
DIM RecordAlimento AS TipoAlimento
DIM TempAlimento AS TipoAlimento
```

Viene aperto il file numero 1 per l'accesso casuale e viene calcolato il numero totale di record in esso contenuti; il generatore di numeri casuali viene inizializzato dall'istruzione:

```
RANDOMIZE TIMER
```

Dal momento che a computer funzionante il timer continua a cambiare, una nuova sequenza di numeri casuali viene generata ad ogni esecuzione del programma.

Un ciclo FOR...NEXT assegna alla variabile k valori interi crescenti da 1 al numero di record del file e ad ogni modifica del valore di k viene assegnato un nuovo numero casuale alla variabile indicecas, quindi un'istruzione GET legge TempAlimento.

```
GET #1, k, TempAlimento
```

Un'altra istruzione GET legge il record di numero indicecas in RecordAlimento.

```
GET #1, indicecas, RecordAlimento
```

Pertanto il record k è copiato in TempAlimento ed il record numero indicecas è copiato in RecordAlimento.

Per riscrivere i record nel file, scambiandone la posizione, vengono utilizzate due istruzioni PUT.

```
PUT #1, k, RecordAlimento
PUT #1, indicecas, TempAlimento
```

Il ciclo si ripete tante volte quanti sono i record del file ed il suo termine coincide con quello del sottoprogramma, che restituisce il controllo al programma principale. Si sostituisca il Programma 9.13 (sottoprogramma MescolaFile) al sottoprogramma MescolaFile temporaneo.

```
REM ** PRO9.13 Sottoprogramma MescolaFile - Usare Visualizza per vedere **

SUB MescolaFile (NomeFile$, NumeroDiRecord%)
' Mescola i record
   ' Inizializza
```

```
  DEFINT A-Z
  DIM RecordAlimento AS TipoAlimento
  DIM TempAlimento AS TipoAlimento
  OPEN NomeFile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)

  ' Calcola il numero dei record e li mescola
  NumeroDiRecord = LOF(1) \ LEN(RecordAlimento)
  RANDOMIZE TIMER
  FOR k = 1 TO NumeroDiRecord
    indicecas = INT(NumeroDiRecord * RND(1)) + 1
    GET #1, k, TempAlimento
    GET #1, indicecas, RecordAlimento
    PUT #1, k, RecordAlimento
    PUT #1, indicecas, TempAlimento
  NEXT k
  CLS
  PRINT "I record ora hanno il seguente ordine:"
END SUB
```

Programma 9.13 Sottoprogramma MescolaFile

9.3.3 Aggiunta del sottoprogramma LeggiRecord

Il programma 9.14, cioè il sottoprogramma LeggiRecord, legge e mostra sullo schermo i record nell'ordine in cui sono correntemente memorizzati nel file; si sostituisca tale sottoprogramma a quello temporaneo e si mandi in esecuzione il Programma 9.11 per verificare tutti e tre i sottoprogrammi: i risultati del test varieranno da un'esecuzione all'altra a causa dell'uso di valori casuali diversi di volta in volta durante il processo di scambio dei record compiuto nel sottoprogramma MescolaFile.

```
REM ** PRO9.14 Sottoprogramma LeggiRecord - Usare Visualizza per vedere **

SUB LeggiRecord (NumeroDiRecord%)
' Legge e visualizza tutti i record
  ' Inizializza
  DEFINT A-Z
  DIM RecordAlimento AS TipoAlimento

 ' Legge i record e ne stampa i campi
  FOR NumeroRecord = 1 TO NumeroDiRecord
    GET #1, NumeroRecord, RecordAlimento
    PRINT RecordAlimento.alimento; RecordAlimento.tipo;
    PRINT RecordAlimento.calorie
  NEXT NumeroRecord
  ' Segnala la fine del file ed attende la pressione di un tasto
  PRINT : PRINT "Fine del file. Premere un tasto per continuare."
```

```
  untasto$ = INPUT$(1)
END SUB
```

Programma 9.14 Sottoprogramma LeggiRecord

9.3.4 Aggiunta del sottoprogramma OrdinaFile

Il programma 9.15 (sottoprogramma OrdinaFile) utilizza le istruzioni GET e PUT per riordinare alfabeticamente il file appena mescolato in base al nome dei cibi. Il ciclo DO...LOOP contiene un semplice algoritmo di ordinamento di tipo Bubblesort, probabilmente la routine di ordinamento più conosciuta per la sua semplicità. Questa routine confronta i record adiacenti e li scambia se sono in ordine sbagliato, facendo si che i record risalgano verso l'alto durante il processo di ordinamento; le istruzioni GET leggono i record adiacenti e, nel caso che questi siano in ordine errato, vengono utilizzate le istruzioni PUT per scambiarne la posizione all'interno del file.

```
REM ** PRO9.15 Sottoprogramma OrdinaFile - Usare Visualizza per vedere **
SUB OrdinaFile (NumeroDiRecord%)

' Mette in ordine alfabetico i record
   ' Inizializza
   DEFINT A-Z
   DIM RecordAlimento AS TipoAlimento
   DIM TempAlimento AS TipoAlimento

   ' Ordina i record
   CLS : PRINT "Ordinamento del file"
   cima = 1
   DO WHILE cima  NumeroDiRecord
     FOR qui = NumeroDiRecord TO cima + 1 STEP -1
       GET #1, qui, TempAlimento
       GET #1, qui - 1, RecordAlimento
       A$ = TempAlimento.alimento: B$ = RecordAlimento.alimento
       IF A$  B$ THEN
         PUT #1, qui, RecordAlimento
         PUT #1, qui - 1, TempAlimento
       END IF
     NEXT qui
     cima = cima + 1
   LOOP
   CLS : PRINT "I record sono stati riordinati:"
END SUB
```

Programma 9.15 Sottoprogramma OrdinaFile

Ciascun carattere di una stringa ha un codice ASCII, ad esempio al carattere A corrisponde il codice ASCII 65 ed alla Z il codice 90, mentre le lettere minuscole hanno codici che

variano da 97 (a) a 122 (z): durante il riordinamento di record contenenti stringhe sono in effetti i codici ASCII ad essere confrontati. In un file mescolato contenente quattro record hanno luogo i seguenti scambi durante il suo ordinamento:

File originale	Primo scambio	Secondo scambio	Terzo scambio	Quarto scambio
Torta	Torta	Avocado	Avocado	Avocado
Banana	Avocado	Torta	Banana	Banana
Avocado	Banana	Banana	Torta	Pane
Pane	Pane	Pane	Pane	Torta

Si sostituisca il nuovo sottoprogramma OrdinaFile a quello temporaneo, si mandi in esecuzione il programma completo (riportato nel Programma 9.16, MESCOLA ED ORDINA UN FILE AD ACCESSO CASUALE FINALE) e lo si salvi con il nome UQB0916.BAS; la Figura 9.7 mostra dopo l'ordinamento il file della Figura 9.6 mescolato.

```
DECLARE SUB LeggiRecord (NumeroDiRecord%)
DECLARE SUB NominaFile (NomeFile$)
DECLARE SUB MescolaFile (NomeFile$, NumeroDiRecord%)
DECLARE SUB OrdinaFile (NumeroDiRecord%)

REM ** MESCOLA E RIORDINA UN FILE AD ACCESSO CASUALE - FINALE **
' Capitolo 9
' Microsoft QuickBASIC 4.5  File: UQB0916.BAS

' Inizializza
CLS : DEFINT A-Z
PRINT "Programma che MESCOLA E RIORDINA UN FILE AD ACCESSO CASUALE"

REM ** DEFINISCE LA STRUTTURA DEL RECORD **
TYPE TipoAlimento
   alimento AS STRING * 18
   tipo AS STRING * 15
   calorie AS INTEGER
END TYPE

' Chiama i sottoprogrammi
CALL NominaFile(NomeFile$)
CALL MescolaFile(NomeFile$, NumeroDiRecord%)

CALL LeggiRecord(NumeroDiRecord%)
CALL OrdinaFile(NumeroDiRecord%)
CALL LeggiRecord(NumeroDiRecord%)

' Chiude il file e termina
CLOSE #1
```

```
END
SUB NominaFile (NomeFile$)
' Assegna un nome al file
   PRINT
   INPUT "Nome del file con estensione"; NomeFile$
   INPUT "Drive contenente il disco"; Drive$
   PRINT : PRINT "Porre il disco con "; NomeFile$; " nel drive "; Drive$
   NomeFile$ = LEFT$(Drive$, 1) + ":" + NomeFile$
   PRINT "Premere un tasto per continuare."
   untasto$ = INPUT$(1)
END SUB

SUB LeggiRecord (NumeroDiRecord%)
' Legge e visualizza tutti i record
   ' Inizializza
   DEFINT A-Z
   DIM RecordAlimento AS TipoAlimento

   ' Legge i record e ne stampa i campi
   FOR NumeroRecord = 1 TO NumeroDiRecord
     GET #1, NumeroRecord, RecordAlimento
     PRINT RecordAlimento.alimento; RecordAlimento.tipo;
     PRINT RecordAlimento.calorie
   NEXT NumeroRecord

   ' Annuncia la fine del file ed attende la pressione di un tasto
   PRINT : PRINT "Fine del file. Premere un tasto per continuare."
   untasto$ = INPUT$(1)
END SUB

SUB MescolaFile (NomeFile$, NumeroDiRecord%)
' Mescola i record
   ' Inizializza
   DEFINT A-Z
   DIM RecordAlimento AS TipoAlimento
   DIM TempAlimento AS TipoAlimento
   OPEN NomeFile$ FOR RANDOM AS #1 LEN = LEN(RecordAlimento)

  ' Calcola il numero dei record e li mescola
   NumeroDiRecord = LOF(1) \ LEN(RecordAlimento)

  RANDOMIZE TIMER
   FOR k = 1 TO NumeroDiRecord
     indicecas = INT(NumeroDiRecord * RND(1)) + 1
     GET #1, k, TempAlimento
     GET #1, indicecas, RecordAlimento
     PUT #1, k, RecordAlimento
     PUT #1, indicecas, TempAlimento
   NEXT k
```

```
   CLS
   PRINT "I record sono ora mescolati nel seguente ordine:"
END SUB
SUB OrdinaFile (NumeroDiRecord%)
' Mette in ordine alfabetico i record
   ' Inizializza
   DEFINT A-Z
   DIM RecordAlimento AS TipoAlimento
   DIM TempAlimento AS TipoAlimento
   ' Ordina i record
   CLS : PRINT "Ordinamento del file"
   cima = 1
   DO WHILE cima  NumeroDiRecord
     FOR qui = NumeroDiRecord TO cima + 1 STEP -1
       GET #1, qui, TempAlimento
       GET #1, qui - 1, RecordAlimento
       A$ = TempAlimento.alimento: B$ = RecordAlimento.alimento
       IF A$ < B$ THEN
         PUT #1, qui, RecordAlimento
         PUT #1, qui - 1, TempAlimento
       END IF
     NEXT qui
     cima = cima + 1
   LOOP
   CLS : PRINT "I record sono stati riordinati:"
END SUB
```

Programma 9.16 MESCOLA ED ORDINA FILE AD ACCESSO CASUALE - FINALE

```
I record sono ora mescolati nel seguente ordine:
Avogado        fresco          48
Insalata       cruda           7
Broccoli       cotti           7
Caciotta       stagionata      30
Banana         fresca          24
Torta          meringata       105
Mirtilli       freschi         18
Biscotti       fatti in casa   105
Carote         crude           12
Mela           fresca          15
Fagioli        lessati         32
Pollo          arrosto         83
Carne          ai ferri        93
```

```
Branzino        fritto              74
Pane            integrale           69
Torta           pane degli angeli   77
Burro           fritto              205
```

Figura 9.6 File non ordinato

```
I record sono stati riordinati:
Avogado         fresco              48
Banana          fresca              24
Biscotti        fatti in casa       105
Branzino        fritto              74
Broccoli        cotti               7
Burro           fritto              205
Carne           ai ferri            93
Carote          crude               12
Caciotta        stagionata          30
Fagioli         lessati             32
Insalata        cruda               7
Mela            fresca              15
Mirtilli        freschi             18
Pane            integrale           69
Pollo           arrosto             83
Torta           meringata           105
Torta           pane degli angeli   77
```

Figura 9.7 File ordinato

9.4 Sommario

Questo capitolo ha mostrato numerosi metodi per creare e manipolare i dati di file sequenziali e ad accesso casuale; i sottoprogrammi sono stati scritti in file separati e riuniti al programma principale con il comando Merge per creare programmi completi in grando

di compiere numerose funzioni. Il Programma 9.5 contiene sottoprogrammi per copiare ed esaminare file sequenziali, il Programma 9.10 sottoprogrammi per aggiungere dati ad un file sequenziale, leggere i dati di un file, copiarli da un file ad un altro, inserire dati tra un record e l'altro ed esaminare un file sequenziale; infine il Programma 9. 16 mescola in ordine casuale i record di un file e li riordina alfabeticamente.

Capitolo 10

Funzioni dinamiche di debug

10.1 Menu Debug
10.2 Analisi del flusso di un programma
10.3 Osservazione di un programma
10.4 Altri strumenti di debug
10.5 Sommario

Spesso gli errori, comunemente chiamati bachi, si fanno strada all'interno dei programmi in modo difficile da scoprire: a volte un programma può apparire buono sulla carta e comportarsi non correttamente al momento dell'esecuzione a causa di un baco elusivo.

Questo capitolo mostra come utilizzare gli strumenti di debug forniti da QuickBASIC per prevenire gli errori, cercare i bachi che impediscono il normale funzionamento dei programmi e trovare anche quelli che causano malfunzionamenti occasionali dei programmi. Per chi avesse già programmato in ambiente GW-BASIC, sono già noti alcuni strumenti di aiuto nella ricerca degli errori di programmazione come le istruzioni STOP, PRINT, CONT, TRON e TROFF; nel caso si sia utilizzata una versione compilata del BASIC, probabilmente ci si sarà serviti di "debugger simbolici", programmi che mettono in grado l'utilizzatore di seguire il flusso di un programma osservandone gli effetti.

QuickBASIC combina le caratteristiche di semplicità del debug del GW-BASIC con i potenti strumenti propri dei debugger simbolici e fornisce strumenti di verifica dei programmi che possono essere utilizzati durante il processo della loro creazione.

Nel capitolo verrà trattato quanto segue.

Analisi del flusso Funzione di QuickBASIC che mostra l'istruzione in corso di esecuzione e comprende comandi per eseguire un'istruzione alla volta, animare l'esecuzione evidenziando ciascuna istruzione al momento della sua esecuzione ed eseguire il programma all'indietro fino ad un massimo di 20 istruzioni dall'ultima raggiunta.

Espressioni di osservazione Espressioni che consentono di osservare il valore di variabili od espressioni durante l'esecuzione di un programma.

Punti di osservazione Espressioni che una volta vere causano l'arresto del programma.

Punti di interruzione Posizioni del programma nelle quali l'esecuzione si arresta temporaneamente per consentire la verifica di porzioni del programma o l'esame del valore di una variabile.

La prevenzione degli errori inizia al momento della progettazione: il migliore metodo per sviluppare programmi senza errori consiste nel ridurre al minimo il numero dei punti nei quali sia possibile che si presentino bachi; nel seguito vengono elencate alcune misure preventive.

- Si progetti il proprio programma con cura prima di sedersi davanti alla tastiera, elencando i compiti che il programma deve svolgere e scrivendolo come un insieme di SUBroutine e FUNCTION che, essendo brevi, sono molto più semplici da correggere di un programma lungo.
- Si utilizzi la finestra Immediata nella parte inferiore dello schermo per isolare e verificare brevi porzioni di codice che, una volta funzionanti, possono essere inserire nel corpo principale del programma.
- Si esegua ciascuna nuova porzione di codice numerose volte all'interno del contesto creato dal resto del programma: in tal modo sarà più semplice trovare errori banali che sarebbe molto più difficile rintracciare nel programma completo.
- Si utilizzino gli strumenti di debug descritti nel resto di questo capitolo.

10.1 Menu Debug

La maggior parte degli strumenti di debug di QuickBASIC sono implementati da comandi del menu Debug, sebbene altri menu di QuickBASIC possano contenere comandi utilizzati durante la ricerca degli errori: i contenuti del menu Debug in modalità semplificata ed in modalità completa sono mostrati nella Tabella 10.1

Tabella 10.1 Menu di Debug in modalità semplificata e completa

Menu semplificato di Debug	Menu completo di Debug
Osserva	Osserva
Osservazione immediata	Osservazione immediata
Elimina dall'osservazione	Punto di osservazione

Menu semplificato di Debug	Menu completo di Debug
Punto di interruzione	Elimina dall'osservazione
Elimina ogni punto interruzione	Elimina ogni osservazione
	Analizza il flusso
	Resoconto
	Punto di interruzione
	Elimina ogni punto interruzione
	Interrompi su errore
	Imposta istruzione successiva

10.2 Analisi del flusso di un programma

L'analisi del flusso del programma mostra quali istruzioni di questo vengano eseguite; a questo scopo QuickBASIC fornisce le funzioni seguenti.

Esecuzione passo passo Premendo il tasto **F8** viene eseguita un'istruzionea la volta.

Esecuzione per procedure Funziona come l'esecuzione passo passo, con l'eccezione che le procedure sono considerate come un'istruzione singola

Analisi del flusso Fornisce un'esecuzione animata evidenziando ciascuna istruzione quando questa riceve il controllo; vengono inoltre registrate le ultime venti istruzioni eseguite per potere essere riviste.

Il programma 10.1, OROLOGIO, cancella lo schermo ed assegna all'orologio di sistema della macchina il valore della variabile inizio, quindi calcola all'interno di un ciclo FOR...NEXT la differenza tra il valore corrente dell'orologio e quello iniziale, assegnando il risultato a TempoTrascorso e visualizzandolo sullo schermo.

```
REM ** OROLOGIO **

' Capitolo 10
' Microsoft QuickBASIC 4.5  File: UQB1001.BAS

REM ** Mostra il trascorrere del tempo **
CLS : DEFINT A-Z
inizio# = TIMER

FOR numero = 1 TO 15
  TempoTrascorso# = TIMER - inizio#
  PRINT numero, TempoTrascorso#
```

```
NEXT numero
END
```

Programma 10.1 OROLOGIO

Si digiti il listato e lo si lanci: durante l'esecuzione si conti fino a dieci e si prema **Ctrl+Break** per arrestarlo; la Figura 10.1 mostra un esempio tipico di output, ma occorre tenere conto del fatto che i tempi trascorsi varieranno a seconda della velocità del computer utilizzato.

```
1          0
2          0
3          0
4          0
5          .0625
6          .0625
7          .0625
8          .0625
9          .0625
10         .0625
11         .0625
12         .11328125
13         .11328125
14         .11328125
15         .11328125

Premere un tasto per continuare
```

Figura 10.1 Output del Programma 10.1 senza utilizzare alcuna funzione di debug

10.2.1 Analisi del flusso

Dopo avere verificato il programma, si apra il menu Debug, si porti l'evidenziazione fino all'opzione Analizza il flusso e si prema **Invio**: apparirà immediatamente la finestra di visualizzazione. Si riapra il menu Debug: come mostrato nella Figura 10.2, vicino alla scritta Analizza il flusso sarà visibile un rombo per indicare che tale funzione è attiva; premendo **Invio** con l'evidenziazione su Analizza il flusso si alterna lo stato della funzione

da attiva (rombo presente) a inattiva (rombo assente). Si esegua il programma con il menu Esegui lasciando attiva l'analisi del flusso: questa funzione di debug rallenta considerevolmente il programma, poiché QuickBASIC esegue le istruzioni passo passo e le evidenzia al momento in cui ricevono il controllo, compiendo una cosiddetta esecuzione animata; essa può essere utilizzata insieme ad altre funzioni di debug descritte più avanti in questo capitolo e consente di determinare il flusso generale del programma in modo da riconoscere se esso si comporta come previsto.

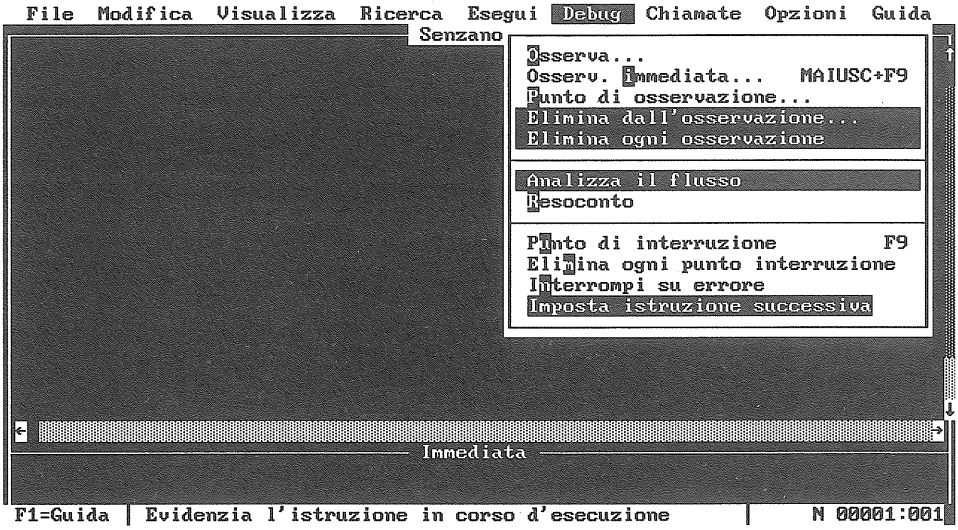


Figura 10.2 Menu Debug con Analizza il flusso evidenziata

Eseguendo il Programma 10.1 con l'animazione vengono immediatamente saltati tutti i commenti (istruzioni REM) e viene evidenziata (in italico) l'istruzione CLS; DEFINT A-Z non è evidenziata perché non è un'istruzione eseguibile.

```
REM ** Mostra il trascorrere del tempo **
CLS : DEFINT A-Z
inizio# = TIMER
FOR numero = 1 TO 15
  TempoTrascorso# = TIMER - inizio#
  PRINT numero, TempoTrascorso#
NEXT numero
```

Subito dopo viene evidenziata l'istruzione inizio# = TIMER:

```
REM ** Mostra il trascorrere del tempo **
CLS : DEFINT A-Z
inizio# = TIMER
FOR numero = 1 TO 15
  TempoTrascorso# = TIMER - inizio#
```

```
  PRINT numero, TempoTrascorso#
NEXT numero
```

Quindi viene evidenziata l'istruzione FOR:

```
REM ** OROLOGIO **
REM ** Mostra il trascorrere del tempo **
CLS : DEFINT A-Z
inizio# = TIMER
FOR numero = 1 TO 15
  TempoTrascorso# = TIMER - inizio#
  PRINT numero, TempoTrascorso#
NEXT numero
```

Le tre istruzioni sono evidenziate a turno 15 volte durante l'esecuzione del ciclo FOR...NEXT:

```
FOR numero = 1 TO 15
  TempoTrascorso# = TIMER - inizio#
  PRINT numero, TempoTrascorso#
NEXT numero

FOR numero = 1 TO 15
  TempoTrascorso# = TIMER - inizio#
  PRINT numero, TempoTrascorso#
NEXT numero

FOR numero = 1 TO 15
  TempoTrascorso# = TIMER - inizio#
  PRINT numero, TempoTrascorso#
NEXT numero
```

L'istruzione END è l'ultima ad essere evidenziata, quindi l'esecuzione termina.

Una volta terminata l'esecuzione il programma resta visibile nella finestra di visualizzazione: per vedere il suo output, si prema il tasto **F4**, che farà apparire la schermata di output, come mostrato nella Figura 10.3.

```
1          .76953125
2          2.41796875
3          4.0078125
4          5.66015625
5          7.25
6          8.8984375
7          10.55078125
8          12.140625
9          13.7890625
```

```
10          15.37890625
11          17.03125
12          18.6796875
13          20.26953125
14          21.91796875
15          23.5703125
```

Figura 10.3 Output del Programma 10.1 con la funzione Analizza il flusso attiva

Confrontando i tempi risultanti con quelli della Figura 10.1, si vedrà che nonostante siano dipendenti dal computer utilizzato il programma rallenta considerevolmente una volta attivata l'analisi del flusso. Si prema **F4** per ritornare alla finestra di visualizzazione.

Si ricordi di premere F4 per passare dalla schermata di output alla finestra di visualizzazione.

Si può verificare la correttezza del flusso del programma osservandolo in esecuzione con l'analisi del flusso attiva; si può inoltre rallentarlo maggiormente servendosi della funzione di esecuzione passo passo.

10.2.2 Esecuzione passo passo

Tramite il comando Esecuzione passo passo si può controllare completamente l'esecuzione del programma. L'esecuzione passo passo verrà sperimentata utilizzando nuovamente il Programma 10.1: invece di avviarlo tramite il menu Esegui, si prema **F8** per evidenziare la sua prima istruzione.

L'esecuzione passo passo viene attivata dal tasto F8, che permette di eseguire un nuovo passo del programma ad ogni sua pressione: innanzitutto viene evidenziata l'istruzione CLS, esattamente come era avvenuto durante l'esecuzione del programma con l'analisi del flusso attiva; l'istruzione evidenziata non viene eseguita fino alla successiva pressione del tasto **F8**, che esegue l'istruzione ed evidenzia la successiva istruzione da eseguire.

Continuando a premere **F8** si osservi come l'evidenziazione si sposti lungo il programma un'istruzione per volta: ciò potrà sembrare tedioso, ma verranno dei momenti nei quali sarà necessario osservare ciascun passo del programma per trovare un errore.

Una volta percorso l'intero programma (inclusa l'istruzione END), si prema **F4** per vedere l'output: si noterà che i valori dei tempi sono cambiati rispetto all'esecuzione precedente: essi sono di gran lunga superiori durante l'esecuzione passo passo; nella Figura 10.4 viene mostrato l'output di una sessione di esecuzione passo passo compiuta premendo il tasto **F8** con un buon ritmo: si noti che questi tempi sono quasi doppi di quelli dell'esecuzione precedente.

```
1          1.859375
2          4.66015625
3          7.30078125
4          9.87890625
5          12.3476525
6          14.76953125
7          17.62890625
8          20.48046875
9          23.390625
10         26.25
11         29
12         31.73828125
13         34.59765625
14         37.55859375
15         40.48046875
```

Figura 10.4 Output del Programma 10.1 utilizzando l'esecuzione passo a passo

10.2.3 Esecuzione per procedure

Per poter dimostrare l'analisi del flusso con l'esecuzione per procedure è necessario disporre di un programma contenente una procedura: si digiti il Programma 10.2, MESCOLA. L'esecuzione per procedure funziona allo stesso modo di quella passo passo, ma esegue le procedure in un unico passo, (ciò viene indicato convenzionalmente con la locuzione "passare sopra la procedura"), rendendo invisibili i singoli passi compiuti all'interno di questa.

```
DECLARE SUB Mescola (array%())
REM ** MESCOLA **
' Capitolo 10
' Microsoft QuickBASIC 4.5  File: UQB1002.BAS

'Dimensiona il vettore di interi numeri
DIM numeri%(1 TO 13)
CLS : DEFINT A-Z

' Crea il vettore e ne mescola gli elementi
FOR k = 1 TO 13
   numeri%(k) = k
```

```
NEXT k
CALL Mescola(numeri%())
END

SUB Mescola (array%())
 ' Mescola il vettore di 13 elementi, quindi
 ' visualizza il vettore ed il tempo di esecuzione
   inizio# = TIMER
   FOR k = 1 TO 13
     indicecas = INT(13 * RND(1)) + 1
     SWAP array%(k), array%(indicecas)
   NEXT k
   TempoTrascorso# = TIMER - inizio#
   FOR k = 1 TO 13
     PRINT array%(k);
   NEXT k
   PRINT : PRINT "Tempo Trascorso ="; TempoTrascorso#
END SUB
```

Programma 10.2 MESCOLA

L'esecuzione per procedure viene attivata dal tasto **F10**. Il suo funzionamento è simile a quello dell'esecuzione passo passo, con l'eccezione che le procedure vengono trattate come passi singoli: vengono eseguiti tutti i passi della procedura, ma sono visibili solamente i risultati della sua chiamata.

Prima di provare l'esecuzione per procedure, si mandi in esecuzione il programma nel solito modo con l'analisi di flusso attiva: il programma contiene una porzione che rileva i tempi, in modo da potere confrontare i tempi necessari a mescolare il vettore utilizzando l'analisi del flusso con quelli richiesti dall'esecuzione passando "sopra" le procedure. Una volta completata l'esecuzione, si prema il tasto **F4** per vedere l'output e lo si annoti per il successivo confronto, quindi si prema **F4** per ritornare alla finestra di visualizzazione. Un tipico risultato con l'analisi del flusso attiva è:

```
9  7   8  5  4  12  10  1  6  3  2  13  11

Tempo Trascorso = 19.94140625
```

Quindi si provi l'esecuzione per procedure: invece di lanciare il programma tramite il menu Esegui, si utilizzi il tasto F10 per percorrere il programma per procedure; il programma principale viene eseguito passo passo, esattamente come nel caso dell'uso del tasto **F8**, fino al raggiungimento della prima istruzione CALL.

```
  FOR k = 1 TO 13
    numeri%(k) = k
  NEXT k

CALL Mescola(numeri%())
  END
```

Premendo **F10** in questo caso verrà eseguito il sottoprogramma e verrà evidenziata l'istruzione END che nel programma principale segue l'istruzione CALL; premendo nuovamente **F10** il programma terminerà. Si esamini l'output premendo il tasto **F4**; un tipico risultato è il seguente:

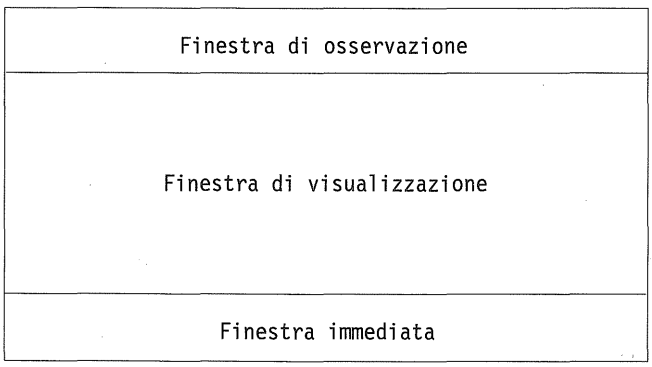
```
9  7  8  5  4  12  10  1  6  3  2  13  11
Tempo Trascorso = .5
```

Si confronti il risultato seguito all'esecuzione precedente con l'analisi del flusso attiva e si noti la differenza dei tempi di esecuzione richiesti per mescolare il vettore, quindi si prema **F4** per ritornare alla finestra di visualizzazione.

Lo stato dell'analisi del flusso (attiva o no) non influisce sul funzionamento passo passo o per procedure; se si desidera disattivare l'analisi del flusso si apra il menu Debug, si evidenzi Analizza il flusso e si prema il tasto **Invio**: se la scritta Analizza il flusso è marcata con un rombo, la pressione del tasto farà scomparire il rombo e disattiverà la funzione, in caso contrario comparirà un rombo prima della scritta e la funzione verrà attivata.

10.3 Osservazione di un programma

Oltre ad analizzare il flusso di un programma, è possibile arrestare la sua esecuzione ad uno specifico punto (di osservazione) per controllare alcuni valori nel corso dell'esecuzione, ferma restando la possibilità di proseguire l'esecuzione da quel punto in modalità passo passo o per procedure. Qualora si utilizzino i comandi Osserva e Punto di osservazione del menu Debug si aprirà una finestra di osservazione nella parte superiore dello schermo, nella quale compariranno i nomi ed i valori di tutte le variabili ed espressioni che si intendano controllare; durante l'esecuzione del programma si possono osservare i cambiamenti dei valori delle espressioni e delle variabili in questa finestra.



Per sperimentare i comandi di osservazione del menu Debug si utilizzi il Programma 10.3, MESCOLA ED ORDINA, simile al Programma 10.2, salvo che il ciclo FOR...NEXT di scrittura dei contenuti dell'array è stato spostato in una subroutine separata e che sono state rimosse tutte le istruzioni di conteggio dei tempi.

Il nome del vettore numerico, dimensionato nel programma principale, condiviso da questo e da tutte le procedure è array%; il vettore viene creato dall'istruzione:

```
DIM SHARED array%(1 TO 13)
```

Segue il listato.

```
DECLARE SUB BubbleSort ()
DECLARE SUB ScriveArray ()
DECLARE SUB Mescola ()
REM ** MESCOLA ED ORDINA **
' Capitolo 10
' Microsoft QuickBASIC 4.5  File: UQB1003.BAS

' Dimensiona il vettore di interi array%
DIM SHARED array%(1 TO 13)
CLS : DEFINT A-Z

' Crea e mescola gli elementi del vettore
FOR k = 1 TO 13
   array%(k) = k
NEXT k

CALL Mescola
CALL ScriveArray
CALL BubbleSort
CALL ScriveArray
END

DEFINT A-Z
SUB BubbleSort
   cima = 1
   DO WHILE cima < 13
     FOR qui = 13 TO cima + 1 STEP -1
       IF array%(qui) < array%(qui - 1) THEN
         SWAP array%(qui), array%(qui - 1)
       END IF
     NEXT qui
     cima = cima + 1
   LOOP
END SUB

SUB ScriveArray
   FOR k2 = 1 TO 13
     PRINT array%(k2)
   NEXT k2
```

```
   PRINT
END SUB

SUB Mescola (array%())
 ' Mescola i 13 elementi del vettore
   punto = 1
   FOR k1 = 1 TO 13
     indicecas = INT(13 * RND(1)) + 1
     SWAP array%(k1), array%(indicecas)
   NEXT k1
   punto = 2
END SUB
```

Programma 10.3 MESCOLA ED ORDINA

10.3.1 Punto di osservazione ed Osserva

I punti di osservazione consentono di sospendere l'esecuzione del programma quando sia verificata una particolare espressione; nel Programma 10.3 viene utilizzato un Bubblesort per ordinare il vettore numerico dopo che questa sia stata mescolata ed i suoi elementi siano stati visualizzati sullo schermo; le istruzioni di chiamata a procedura sono inizialmente:

```
CALL Mescola
CALL ScriveArray

CALL BubbleSort
CALL ScriveArray
```

Il sottoprogramma ScriveArray viene chiamato sia dopo che i numeri sono stati mescolati che dopo l'esecuzione del sottoprogramma di ordinamento, in modo da visualizzare sia il vettore mescolato che quello finale ordinato; si noti che nella subroutine Mescola non viene utilizzata nessuna istruzione RANDOMIZE per inizializzare il generatore di numeri casuali, per cui la variabile indicecas riceverà sempre la stessa sequenza di numeri.

Essendo questo programma più lungo del Programma 10.2, la sua esecuzione passo passo richiederebbe un tempo eccessivo: si mandi in esecuzione il programma dopo avere attivato l'analisi del flusso e si utilizzi un punto di osservazione per sospendere l'esecuzione all'ingresso nel sottoprogramma Mescola, servendosi di un'espressione che sia falsa fino al raggiungimento della prima istruzione eseguibile del sottoprogramma:

```
SUB Mescola
' Mescola i 13 elementi del vettore
  punto = 1                          <Prima istruzione eseguibile del programma
  FOR k = 1 TO 13
  .
  .
```

L'esecuzione del programma verrà sospesa quando l'espressione sarà divenuta vera: a questo punto si esegua passo passo Mescola premendo il tasto **F8**.

È possibile osservare gli scambi dei numeri del vettore aggiungendo opportune espressioni di osservazione; si utilizzi il tasto **F10** per eseguire per procedure il resto del programma una volta completata l'esecuzione passo passo della procedura Mescola.

Si digiti il Programma 10.3 e lo si salvi in formato testo con il nome UQB1003.BAS, quindi si apra il menu Debug, si porti l'evidenziazione su Analizza il flusso e si prema **Invio**.

Si porti nella finestra di visualizzazione la subroutine Mescola utilizzando il menu Visualizza, si apra nuovamente il menu Debug, si porti l'evidenziazione su Punto di osservazione e si prema **Invio**: apparirà un riquadro, come mostrato nella Figura 10.5.

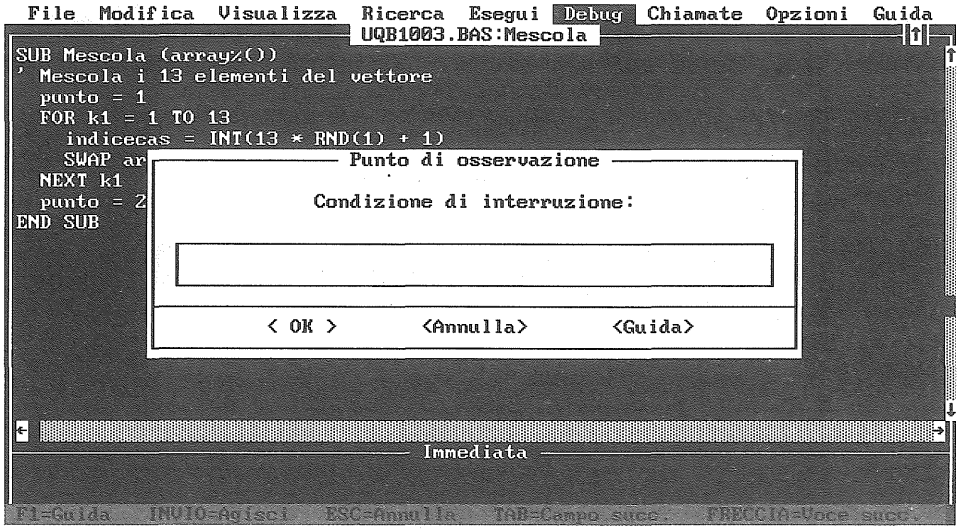


Figura 10.5 Riquadro di Punto di osservazione

Si digiti **punto = 1** come espressione da osservare e si prema **Invio**: verrà aperta una finestra di osservazione contenente il punto di osservazione appena impostato sopra quella di visualizzazione.

```
Mescola punto = 1; <FALSO>
```

Come indicato nella finestra di osservazione, il punto di osservazione (punto = 1) è falso dal momento che il programma non è stato ancora avviato.

Potrebbe essere interessante osservare durante l'esecuzione della procedura Mescola l'evoluzione dei valori delle variabili k1 ed indicecas e degli elementi del vettore array%(k1)

ed array%(indicecas); per inserire queste variabili nella finestra di osservazione, se non fosse già stato fatto si mostri la subroutine Mescola nella finestra di visualizzazione, quindi si apra il menu Debug e si selezioni il comando Osserva: premendo **Invio** apparirà il riquadro mostrato nella Figura 10.6.

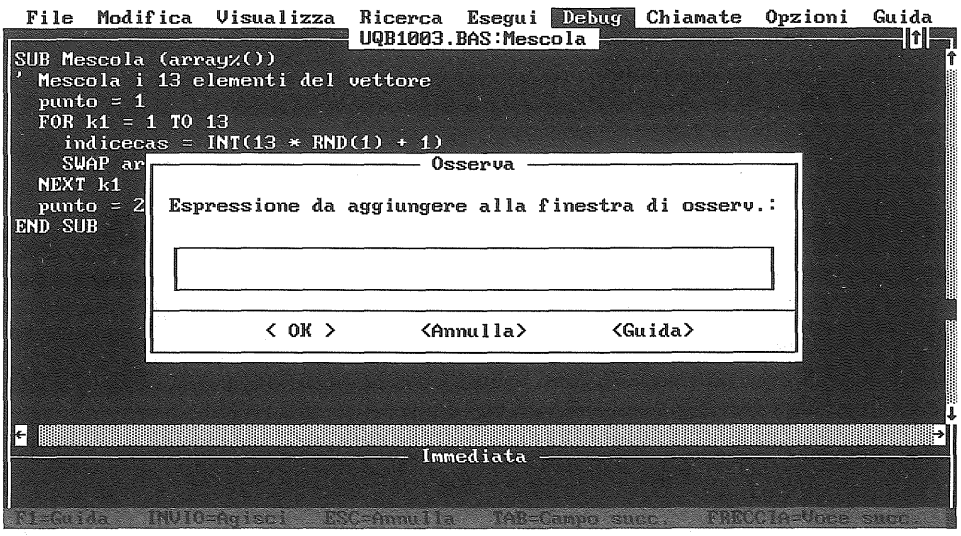


Figura 10.6 Riquadro Osserva

Si digiti **k1** e si prema **Invio** per aggiungere tale espressione alla finestra di osservazione: si tornerà alla finestra di visualizzazione e la variabile k1 sarà visibile nella finestra di osservazione nella parte superiore dello schermo immediatamente sotto il punto di osservazione punto = 1.

```
Mescola punto = 1; <FALSO>
Mescola k1:
```

Si impartisca nuovamente il comando Osserva del menu debug e si digiti indicecas come nome dell'espressione da aggiungere alla finestra di osservazione; a questo punto si avrà la situazione seguente:

```
Mescola punto = 1; <FALSO>
Mescola k1:
Mescola indicecas:
```

Si apra il menu Debug altre due volte per aggiungere le variabili del vettore array%(k1) ed array%(indicecas), portando così a quattro il numero delle variabili in osservazione (oltre al punto di osservazione stabilito in precedenza). Si possono osservare i cambiamenti dei valori di queste variabili durante l'esecuzione di ciascun passo della procedura Mescola; la finestra di osservazione conterrà i seguenti elementi:

```
Mescola punto = 1; <FALSO>
Mescola k1:
Mescola indicecas:
Mescola array%(k1):
Mescola array%(indicecas):
```

Si attivi la funzione Analizza il flusso del menu Debug prima di eseguire il programma, quindi si apra il menu Esegui, si evidenzi il comando Avvia e si prema **Invio**, mantenendo lo sguardo rivolto alla finestra di osservazione durante l'esecuzione del programma. Inizialmente il punto di osservazione punto = 1 è FALSO, mentre le quattro variabili del sottoprogramma Mescola presentano la scritta "Non osservabile" dal momento che questo non è stato ancora raggiunto ed i loro valori sono per il momento indeterminati.

I punti di osservazione consentono di passare velocemente attraverso un programma fino al raggiungimento di una posizione predeterminata, dalla quale si può proseguire l'esecuzione procedendo passo passo attraverso la porzione seguente per verificare i risultati delle singole istruzioni.

Il programma scorre velocemente lungo le varie istruzioni fino ad eseguire l'istruzione punto = 1 del sottoprogramma Mescola, che rendendo vera l'espressione punto = 1 della finestra di osservazione sospende l'esecuzione del programma.

```
Mescola K1 = 1: <vero>
Mescola k1:  0
Mescola indicecas:  0
Mescola array%(k1): Indice fuori dai limiti
Mescola array%(indicecas): Indice fuori dai limiti
```

L'istruzione FOR è evidenziata: si utilizzi il tasto **F8** per eseguire il sottoprogramma MESCOLA passo passo e si osservi come gli elementi del vettore siano scambiati tra di loro quando indicecas è diverso da k1: i primi due scambi avvengono come mostrato nella Tabella 10.2.

Se k1 ed indicecas hanno gli stessi valori, il valore di array%(k1) non cambia ed il valore di k1 viene incrementato di uno prima di compiere un altro ciclo. La sequenza di numeri casuali assegnata alla variabile indicecas è 10, 7, 8, 4, 4, 11, 1, 10, 1, 6, 12 e viene generata

ad ogni esecuzione del programma a meno che non si inserisca un'istruzione RANDOMIZE (di cui si è parlato precedentemente) prima dell'esecuzione del ciclo FOR...NEXT.

Tabella 10.2 Osservazione delle variabili durante i primi due scambi

Primo passo del ciclo FOR...NEXT

	Istruzione evidenziata		
Variabile osservata	indicecas	SWAP	NEXT k1
k1	1	1	1
indicecas	0	10	10
array%(k1)	1	1	10
array%(indicecas)	0	10	1

Secondo passo del ciclo FOR...NEXT

	Istruzione evidenziata		
Variabile osservata	indicecas	SWAP	NEXT k1
k1	2	2	2
indicecas	10	7	7
array%(k1)	2	2	7
array%(indicecas)	1	7	2

La Tabella 10.3 mostra i valori degli elementi di array%(k1) dopo dopo l'esecuzione dell'istruzione SWAP ad ogni passo del ciclo; gli elementi del vettore scambiati sono evidenziati. Una volta usciti dal sottoprogramma mescola si continui a premere il tasto **F10** per eseguire i sottoprogrammi BubbleSort e ScriveArray: verrà visualizzato il vettore finale ordinato; premendo nuovamente il tasto **F10** il programma terminerà. Si prema **F4** per accedere alla schermata di Output, nella quale saranno visibili il vettore originale mescolato e quello finale ordinato.

```
9  7  8  5  4  12  10  1  6  3  2  13  11
1  2  3  4  5  6  7  8  9  10  11  12  13
```

Per disattivare i punti di osservazione e le espressioni di osservazione si apra il menu Debug, si selezioni Elimina ogni osservazione e si prema **Invio**; per disattivare l'analisi del flusso, si riapra il menu Debug si porti l'evidenziazione su Analizza il flusso e si prema il tasto **Invio**.

Tabella 10.3 Elementi di array% dopo gli scambi

	Elementi nel vettore												
Swap	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	***10***	2	3	4	5	6	7	8	9	***1***	11	12	13
22	10	***7***	***8***	4	5	6	***2***	8	9	1	11	12	13
3	10	7	8	4	5	6	2	***3***	9	1	11	12	13
4	10	7	8	4	5	6	2	3	9	1	11	12	13
5	10	7	8	***5***	***4***	6	2	3	9	1	11	12	13
6	10	7	8	5	4	***11***	2	3	9	1	***6***	12	13
7	***2***	7	8	5	4	11	***10***	3	9	1	6	12	13
8	2	7	8	5	4	11	10	***1***	9	***3***	6	12	13
9	2	7	8	5	4	11	10	1	***6***	3	***9***	12	13
10	2	7	8	5	4	11	10	1	6	3	9	12	13
11	***9***	7	8	5	4	11	10	1	6	3	***2***	12	13
12	9	7	8	5	4	***12***	10	1	6	3	2	***11***	13
13	9	7	8	5	4	12	10	1	6	3	2	***13***	***11***

10.3.2 Punto di interruzione ed osservazione immediata

Un punto di interruzione è una posizione del programma alla quale si arresta l'esecuzione: queste posizioni possono essere impostate servendosi del comando Punto di interruzione del menu Debug o premendo il tasto **F9** senza aprire il menu Debug.

I punti di interruzione possono essere utilizzati per verificare porzioni individuali del programma ed inseriti in posizioni strategiche per arrestare un programma in modo da potere esaminare il valore assunto in quel momento da una variabile: ciò li rende un valido strumento in grado di localizzare i punti di un programma che causano errori durante l'esecuzione.

Il comando Osservazione immediata visualizza il valore di una variabile od espressione (vero o falso) quando viene sospesa l'esecuzione di un programma ed è utile nel caso non sia necessario controllare una variabile con continuità, ma solo in alcuni momenti nei quali il programma non sia più in esecuzione, come i punti di interruzione. L'utilizzo dei punti di interruzione e dell'osservazione immediata può essere illustrato introducendo volutamente un errore nel Programma 10.3: si visualizzi nell'apposita finestra il sottoprogramma BubbleSort e si porti il cursore sull'istruzione:

```
cima = cima + 1
```

Modificando la parte destra dell'assegnamento in:

```
cima = cima + 2
```

Si mandi in esecuzione il Programma 10.3 con questa modifica e si finga di non sapere che l'istruzione sopra riportata contiene un errore; i numeri saranno mescolati e visualizzati nell'ordine seguente:

```
9  7  8  5  4  12  10  1  6  3  2  13  11
1  9  2  7  3  8  4  5  6  10  11  12  13
```

I numeri non sono ordinati correttamente, quindi il sottoprogramma BubbleSort non funziona come previsto. Per cercare l'errore in questa porzione del programma si visualizzi il sottoprogramma BubbleSort per mezzo del menu Visualizza, quindi si porti il cursore alla posizione nella quale si intende inserire il punto di interruzione, ad esempio sull'istruzione LOOP al termine del ciclo DO...LOOP. Con il cursore in questa posizione si apra il menu Debug, si porti l'evidenziazione su Punto di interruzione e si prema **Invio**: lo sfondo dell'intera linea contenente l'istruzione LOOP cambierà colore.

```
    cima = cima + 1
  LOOP
END SUB
```

In questo modo è stato appena inserito un punto di interruzione nel sottoprogramma BubbleSort. Si esegua nuovamente il Programma 10.3, che questa volta si arresterà al termine del DO...LOOP, sempre con lo sfondo della linea contenente l'istruzione LOOP del colore del punto di interruzione. A questo punto è stata eseguita una passata del ciclo DO...LOOP: poiché cima ha ricevuto il valore 1 prima di entrare nel ciclo, ora dovrebbe avere il valore 2; è il momento di utilizzare il comando Osservazione immediata. Si porti il cursore sulla prima lettera della variabile cima:

```
  cima = cima + 2
LOOP
```

Tenendo premuto il tasto **Shift** si prema **Freccia Destra** quattro volte per evidenziare la parola cima, quindi si apra il menu Debug, si porti l'evidenziazione su Osservazione immediata e si prema **Invio**: si verrà riportati alla finestra di visualizzazione contenente la SUBroutine BubbleSort, ma apparirà anche il riquadro mostrato nella Figura 10.7 indicante che il valore di cima è 3. Il comando Osservazione immediata visualizza il valore corrente dell'espressione o della variabile selezionata in un riquadro, per rimuovere il quale occorre

premere **Esc**. Si prema quindi **Freccia Sinistra** una volta per togliere l'evidenziazione dalla parola cima. A questo punto si tolga il punto di interruzione seguendo i passi elencati:

- si provi il cursore all'inizio della parola LOOP;
- si apra il menu Debug;
- si porti l'evidenziazione su Punto di interruzione;
- si prema **Invio**.

Il programma riporterà alla finestra di visualizzazione dalla quale sarà stato tolto il punto di interruzione in corrispondenza dell'istruzione LOOP. A questo punto si inserisca un punto di interruzione nell'istruzione:

```
cima = cima + 2
```

Si utilizzi il procedimento seguito in precedenza:

- si apra il menu Debug con il cursore sulla lettera c di cima;
- si selezioni Punto di interruzione;
- si prema **Invio**.

Si esegua nuovamente il programma una volta evidenziata l'istruzione selezionata, alla quale si arresterà l'esecuzione. Si evidenzi la parola cima premendo quattro volte il tasto **Freccia Destra** contemporaneamente al tasto **Shift**, si apra il menu Debug, si evidenzi Osservazione istantanea e si prema **Invio**: apparirà un riquadro di comunicazione simile a quello mostrato nella Figura 10.7, con la differenza che il valore della variabile cima sarà questa volta 1 (prima dell'esecuzione dell'istruzione); si prema **Esc** per ritornare alla finestra di visualizzazione. Continuando l'esecuzione del programma, questa proseguirà da questo punto fino a raggiungere nuovamente l'istruzione evidenziata e si potrà vedere che il valore di cima è cambiato; si apra il menu Esegui, si porti l'evidenziazione su Continua e si prema **Invio** per proseguire l'esecuzione da questo punto e fermarsi nuovamente sull'istruzione di assegnamento del valore di cima. Si evidenzi nuovamente la parola cima e si richiami il comando Osservazione istantanea del menu Debug: premendo **Invio** apparirà nuovamente il riquadro di comunicazione, che mostrerà come il valore di cima sia cambiato da 1 a 3, incrementando di 2 invece che di 1: l'errore sarà trovato osservando l'istruzione di assegnamento.

10.4 Altri strumenti di debug

Il menu Debug fornisce altri strumenti utili per la localizzazione degli errori difficili da rilevare, come Resoconto, Interrompi su errore ed Imposta istruzione successiva.

10.4.1 Resoconto

Questo strumento è utile per controllare il flusso del programma, particolarmente nel caso si debba seguire il programma durante l'esecuzione di istruzioni comprendenti salti condizionati. Quando l'opzione Resoconto del menu Debug è preceduta da un rombo è possibile vedere l'effetto degli ultimi 20 passi eseguiti servendosi della combinazione di tasti di resoconto all'indietro, **Shift+F8**, ad ogni pressione dei quali viene compiuto un passo all'indietro nell'esecuzione del programma.

È anche possibile, dopo avere percorso all'indietro una sequenza di istruzioni, utilizzare la combinazione di tasti di resoconto in avanti, **Shift+F10** per ripercorre in avanti la stessa sequenza, eseguendo un'istruzione ad ogni pressione di **Shift+F10**. Si apra il menu Debug, si porti l'evidenziazione sulla scritta Resoconto e si prema **Invio**, quindi si riapra il menu Debug e si inserisca un punto di interruzione in corrispondenza dell'istruzione LOOP del sottoprogramma BubbleSort.

Si mandi nuovamente in esecuzione il Programma 10.3 e, all'arresto dell'esecuzione all'istruzione LOOP, si prema **Shift+F8** 20 volte osservando il cursore muoversi all'indietro lungo il ciclo; nel caso si provi ad andare indietro di più di 20 istruzioni verrà emessa una nota.

A questo punto si prema **Shift+F10** per muoversi avanti di un passo per ogni pressione di tasto: si noti che in questa porzione di programma il flusso del programma appare corretto, dal che si può desumere che l'errore non è dovuto all'ordine delle istruzioni.

*La funzione Resoconto viene disattivata premendo **Invio** con la scritta Resoconto evidenziata e preceduta da un rombo.*

10.4.2 Interrompi su errori

L'opzione Interrompi su errori è un comando avanzato di debug che consente di rintracciare un'istruzione che causi errori in un programma che utilizzi le istruzioni di gestione degli errori; essa non viene trattata in questo libro, per cui, se si intendesse utilizzarla, si faccia riferimento al manuale Microsoft.

10.4.3 Imposta l'istruzione successiva

Imposta l'istruzione successiva è un altro strumento avanzato di debug che consente di saltare una sezione di programma al momento del suo riavvio: una volta arrestata l'esecuzione del programma, si porti il cursore su una nuova istruzione e si richiami Imposta

l'istruzione successiva: questo procedimento altera il flusso di esecuzione sostituendo alll'istruzione successiva quella indicata dal cursore. Questo comando deve essere utilizzato con cautela: saltare le istruzioni può lasciare indefinito il valore di alcune variabili e provocare errori; si noti che l'effetto di Imposta l'istruzione successiva è simile a quello dell'istruzione GOTO.

Per verificare il corretto funzionamento dei programmi e localizzare i bachi che altrimenti sarebbero difficili da trovare si possono utilizzare varie combinazioni delle funzioni di debug discusse in questo capitolo.

10.5 Sommario

QuickBASIC contiene numerosi controlli per rilevare gli errori di programmazione banali e dispone di alcuni strumenti di debug per trovare i bachi più elusivi, strumenti che costituiscono l'oggetto della trattazione di questo capitolo che ha discusso i comandi presenti nel menu Debug e ne ha dimostrato l'utilizzo.

Si è appreso come attivare l'analisi del flusso con l'esecuzione passo passo e per procedure e ad utilizzare espressioni e punti di osservazione.

È stato anche illustrato l'utilizzo dei comandi Punto di interruzione ed Osservazione immediata, insieme alla funzione Resoconto ed alle combinazioni di tasti ad essa correlate per muoversi indietro ed avanti lungo il flusso del programma.

Capitolo 11

File eseguibili

11.1 Creazione di file eseguibili
11.2 Creazione di un file .EXE che richiede BRUN45.EXE
11.3 Creazione di un file .EXE autonomo
11.4 Sommario

Mentre nei precedenti capitoli sono state presentate le caratteristiche di base di QuickBASIC, in questo capitolo verranno presentati i per creare file contenenti programmi QuickBASIC che possano essere eseguiti al di fuori del suo ambiente integrato; i programmi di questo capitolo forniscono inoltre una breve introduzione alla grafica su computer illustrando come tracciare grafici a barre ed a torta.

Tutti i programmi QuickBasic presentati nei capitoli precedenti sono stati creati e salvati in modo da potere essere eseguiti solamente dal disco di lavoro di QuickBASIC, ma, una volta realizzato un programma eseguibile nell'ambiente integrato, se ne può creare una versione funzionante richiamabile direttamente dalla linea di comando del'DOS: tale versione viene chiamata file eseguibile.

I file eseguibili possono essere mandati in esecuzione dal MS-DOS digitandone il nome e sono ottimizzati in velocità ed efficienza, per cui eseguono l'aritmetica in virgola mobile molto più velocemente ed efficientemente degli stessi programmi eseguiti in ambiente QuickBASIC.

11.1 Creazione di file eseguibili

I file eseguibili hanno l'estensione .EXE: ad esempio, se è stato realizzato un programma QuickBASIC che è stato memorizzato con il nome UQB1101.BAS, il file eseguibile creato a partire da quel programma avrà il nome UQB1101.EXE. Una volta posto il dischetto contenente il file eseguibile (UQB1101.EXE) nell'unità B si potrà accedere al DOS

dall'unità A utilizzando il proprio dischetto MS-DOS, oppure il proprio dischetto di lavoro QuickBASIC (se contiene COMMAND.COM) e digitare quindi il seguente comando:

```
A>B:UQB1101
```

Nel caso si intenda creare un nuovo dischetto di lavoro QuickBASIC, basterà che questo contenga i file seguenti.

COMMAND.COM dal dischetto di sistema MS-DOS.

QB.EXE dal dischetto QuickBASIC.

QB45QCK.HLP dal dischetto QuickBASIC.

Ci si assicuri della disponibilità dei file appropriati prima di iniziare il processo di creazione di un file eseguibile; se non si possiede un disco fisso su cui sono presenti tutti i file di QuickBASIC e si utilizzano drive per dischetti da 5"1/4 si dovranno copiare i propri file ed altri programmi su numerosi floppy disk.

Nello sviluppo dei programmi di questo libro sono stati utilizzati due sistemi differenti, nessuno dei quali dotato di disco fisso: il primo è stato un sistema "ridotto all'osso" che risponde ai requisiti minimi per utilizzare QuickBASIC: 384K di memoria e due drive per dischetti da 5"1/4 (aventi capacità di 360K ciascuno).

Il secondo sistema è leggermente più grande e costituisce una via di mezzo tra un sistema minimale ed uno a disco fisso: è dotato di 640K di memoria, un drive per dischetti da 5"1/4 (da 360K) ed un drive per dischetti da 3"1/2 (da 720K di capacità). I sistemi dotati di disco fisso possono memorizzarvi tutti i file di QuickBASIC, pronti per essere richiamati in qualsiasi momento, mentre un sistema con soli drive per dischetti da 5"1/4 richiede numerosi floppy disk e la sostituzione dei dischetti quando necessario; un sistema con un drive per dischetti da 3"1/2 da 720K può memorizzare i file richiesti per numerose funzioni di QuickBASIC, tuttavia per alcuni compiti potrebbe essere necessario un secondo disco. La descrizione che segue è basata su un sistema con un drive per dischetti da 3"1/2 (720K) ed un drive per floppy disk da 5"1/4 (360K). Quando, durante la creazione di un file eseguibile, QuickBASIC non è in grado di trovare un file necessario al processo di creazione, viene richiesto all'utente di digitare un percorso: in questo caso si inserisca il dischetto appropriato nel drive e si risponda al prompt; è anche possibile impostare i percorsi per localizzare i vari file utilizzando il menu Opzioni, come si vedrà nel seguito.

L'utilizzo del dischetto di lavoro QuickBASIC non dà problemi fino a quando si lavori solamente con programmi eseguibili nell'ambiente QuickBASIC; purtroppo la creazione di file eseguibili richiede ulteriori file, descritti nel manuale Microsoft.

11.1.1 Preparazione dei dischetti di lavoro

Esistono numerosi metodi per organizzare i file aggiuntivi in un sistema a dischetti; le configurazioni mostrate nella Tabella 11.1 sono state utilizzare per creare i file .EXE di

questo capitolo; in particolare le configurazioni della colonna destra della tabella richiedono l'utilizzo di tre dischetti di lavoro.

Tabella 11.1 Configurazioni dei file .EXE

Sistema dotato di due drive da 5"1/4	Sistema dotato di un drive da 3"1/2 ed uno da 5"1/4
1. Disco di lavoro QB	**1. Disco di lavoro QB (3"1/2)**
COMMAND.COM	COMMAND.COM
QB.EXE	QB.EXE
QB45.HLP	QB45.HLP
	BC.EXE
	LINK.EXE
2. Disco di compilazione	**2. Disco di libreria (5"1/4)**
BC.EXE	BRUN45.LIB
LINK.EXE	BCOM45.LIB
3. Disco di libreria	**3. Disco dei programmi (3"1/2)**
BRUN45.LIB	COMMAND.COM
BCOM45.LIB	BRUN45.EXE
4. Disco dei programmi	file .EXE che richiedono BRUN45.EXE
COMMAND.COM	
BRUN45.EXE	
file .EXE che richiedono BRUN45.EXE	

Durante la compilazione si utilizzerà il proprio disco di lavoro QuickBASIC come disco principale: nel caso si tratti di un disco da 3 pollici e 1/2, i file aggiuntivi da copiare dai dischi Microsoft su questo disco sono i seguenti.

BC.EXE Crea file oggetto collegabili (.OBJ) a partire dal programma QuickBASIC

LINK.EXE Collega i file oggetto creati da BC.EXE.

Un tipico elenco dei file del dischetto di lavoro QuickBASIC comprenderà i seguenti file:

```
COMMAND  COM
QB       EXE
QB45QCK  HLP
BC       EXE
LINK     EXE
```

Il secondo disco utilizzato, il disco di libreria, dovrebbe essere creato a partire da un disco da 5"1/4 su cui copiare dai dischetti Microsoft i due seguenti file:

```
BRUN45.LIB
BCOM45.LIB
```

Il file BRUN45.LIB viene utilizzato durante la creazione di file eseguibili che richiedono BRUN45.EXE, mentre il file BCOM45.LIB viene utilizzato durante la creazione di file eseguibili autonomi.

Il terzo disco di lavoro è il disco dei programmi, che deve essere creato formattando un dischetto da 3"1/2 con i file di sistema ed il file COMMAND.COM del DOS e conterrà i file .EXE prodotti a partire di file .BAS.

I programmi eseguibili potranno essere lanciati direttamente dal DOS servendosi solamente di questo dischetto.

Si copi il file BRUN45.EXE da uno dei dischetti di utilità di QuickBASIC nel disco dei programmi: questo file è necessario per mandare in esecuzione i file eseguibili non autonomi.

Seguendo questo metodo sono stati preparati due dischetti da utilizzare nel processo di creazione dei file .EXE ed un dischetto per eseguire i file .EXE creati: il dischetto di lavoro da 3"1/2 sarà utilizzato nell'unità A per creare i file .EXE, il floppy da 5"1/4 sarà posto nell'unità B per fornire i file di libreria necessari durante il processo ed infine il secondo dischetto da 3"1/2 sarà sostituito alla fine per eseguire il file .EXE creato.

11.1.2 Scelta dei percorsi

Il menu Opzioni completo consente di scegliere i percorsi di ricerca dei vari tipi di file di QuickBASIC, che dovrebbero essere impostati correttamente per evitare interruzioni durante il processo di creazione dei file eseguibili.

La Figura 11.1 mostra i percorsi impostati per le dimostrazioni di questo capitolo, che, dal momento che non sono state utilizzate directory nelle dimostrazioni, comprendono solo l'unità su cui risiedono i file; nel caso si utilizzino directory o sottodirectory si aggiungano i nomi delle directory e sottodirectory appropriate nei percorsi impostati.

11.1.3 Meccanismo di creazione dei file .EXE

Durante la creazione di un file eseguibile, QuickBASIC si serve innanzitutto di BC.EXE (il compilatore BASIC) per compilare il programma creando un file intermedio noto come file oggetto, quindi il programma LINK.EXE collega tutti i file oggetto creati da BC.EXE

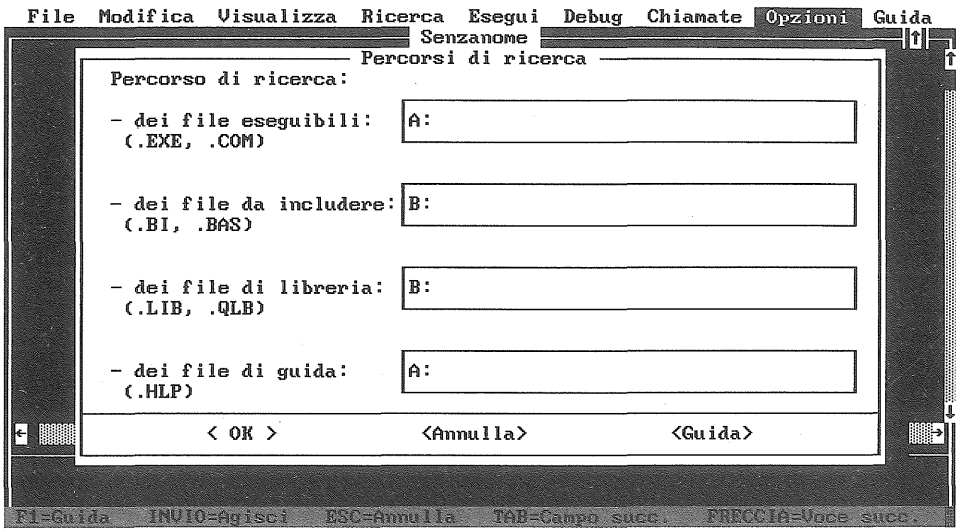
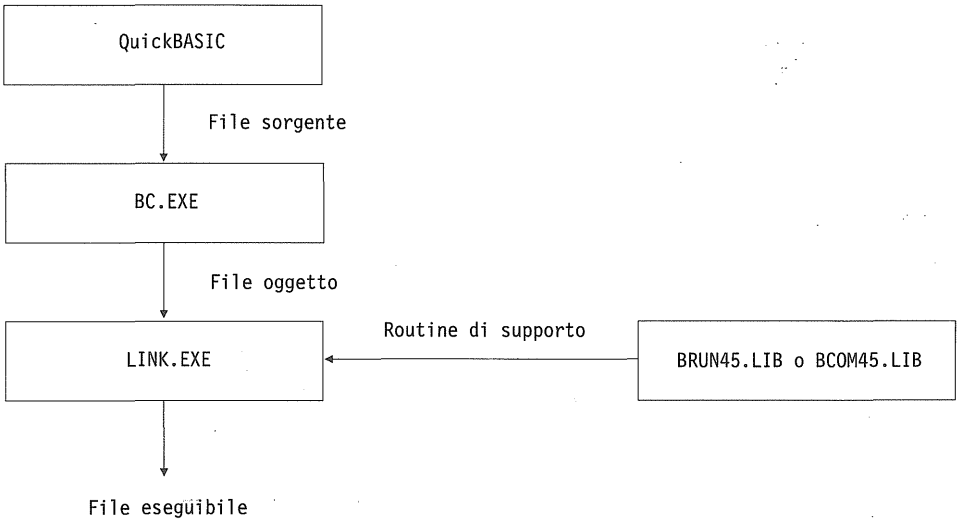


Figura 11.1 Riquadro di scelta dei percorsi

(compilando separatamente i moduli che compongono il programma) in un solo file eseguibile combinandoli anche con le routine di supporto necessarie che si trovano nelle librerie di run-time di BRUN45.LIB e BCOM45.LIB.

Di seguito viene mostrato un diagramma semplificato di questo processo.



QuickBASIC può creare file eseguibili di due tipi:

- file .EXE che richiedono BRUN45.EXE;
- file .EXE autonomi.

La scelta del tipo viene compiuta caricando il programma QuickBASIC e scegliendo il comando Crea un file EXE del menu Esegui. Nel seguito verrà discusso l'intero processo di creazione di un file eseguibile: per ora si apra il menu Esegui e si selezioni il comando Crea un file EXE per prendere in esame il riquadro di comunicazione mostrato nella Figura 11.2

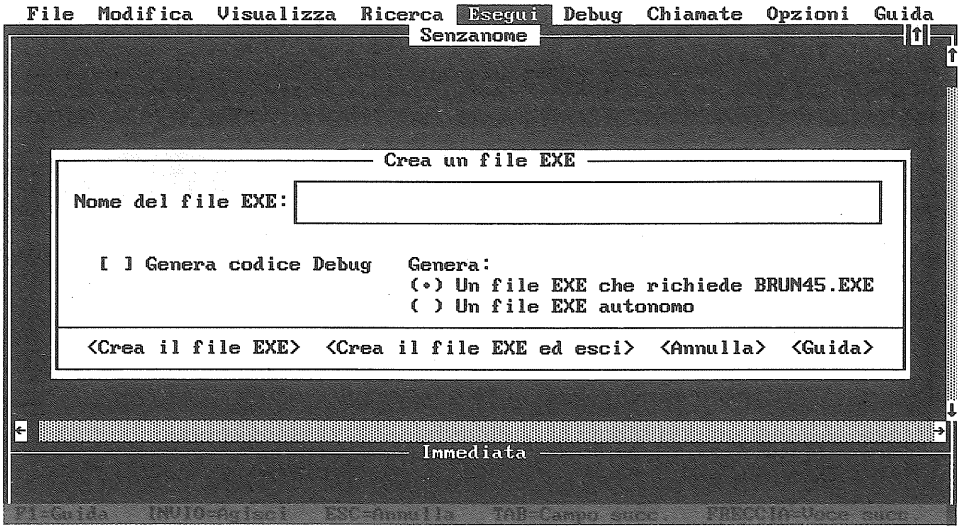


Figura 11.2 Riquadro di creazione dei file .EXE

Il riquadro Nome del file EXE è vuoto ad eccezione del cursore poiché non è stato caricato in memoria nessun file QuickBASIC; se un programma fosse presente in memoria sarebbe riportato il suo nome con l'aggiunta dell'estensione .EXE. Si prema una volta il tasto **Tab** ed il cursore si sposterà tra le parentesi quadre alla sinistra di Genera codice Debug.

```
[_] Genera codice Debug
```

Si digiti **D** e tra le parentesi apparirà una X, indicando che la funzione di generazione del codice di debug è stata attivata.

```
[X] Genera codice Debug
```

Si digiti nuovamente **D** per togliere la X: come si sarà notato il tasto **D** serve sia ad attivare che a disattivare questa opzione.

Selezionando l'opzione Genera codice Debug il codice prodotto al momento dell'esecuzione controllerà:

- l'overflow ed underflow aritmetici;
- gli indici e limiti delle matrici;
- la posizione delle linee nel codice, in modo da potere indicare nei messaggi di errore il numero della linea che lo ha generato;
- le istruzioni RETURN, verificando che corrispondano ai GOSUB;
- se dopo l'esecuzione di ogni linea sia stato premuto **Ctrl+Break**, nel qual caso viene arrestata l'esecuzione.

Se si desidera che il programma risponda alla combinazione di tasti **Ctrl+Break** è necessario compilare il programma con la funzione Genera codice Debug attiva, altrimenti **Ctrl+Break** arresterà il programma solo in una delle seguenti circostanze:

- immissione di dati in risposta al prompt di un'istruzione INPUT;
- stesura espressamente studiata in modo da controllare la pressione di **Ctrl+Break**.

Si prema nuovamente il tasto **Tab** per spostare il cursore da Genera codice Debug all'area indicata da Genera:, che contiene due scelte, la prima delle quali è marcata da un rombo posto tra le parentesi, indicando che attualmente è selezionato un file .EXE che richiede BRUN45.EXE.

```
Genera:
(♦) Un file EXE che richiede BRUN45.EXE
( ) Un file EXE autonomo
```

Questa scelta rende necessaria la presenza del file speciale BRUN45.EXE insieme al file .EXE creato per rendere possibile la sua esecuzione.

Si prema il tasto **Freccia Giù** per spostare il rombo sulla seconda scelta.

```
Genera:
( ) Un file EXE che richiede BRUN45.EXE
(♦) Un file EXE autonomo
```

Se viene attivata questa scelta il file .EXE prodotto sarà eseguibile senza il file speciale (BRUN45.EXE).

Si premano i tasti freccia per spostare il rombo da una scelta all'altra.

Nella parte inferiore del riquadro sono visibili quattro pulsanti: si prema il tasto **Tab** per spostarsi dall'area Genera: al pulsante più a sinistra, Crea il file EXE, che è evidenziato; se questo pulsante è evidenziato al momento della pressione del tasto **Invio** il file viene compilato, viene creato il file .EXE e si ritorna all'editor di QuickBASIC.

Si prema nuovamente il tasto **Tab** per passare al pulsante successivo, Crea il file EXE ed esci: premendo il tasto Invio con questo pulsante evidenziato il file viene compilato, viene creato il file .EXE e viene abbandonato QuickBASIC.

Un ulteriore pressione del tasto **Tab** passa al pulsante successivo, Annulla: premendo Invio con questo pulsante evidenziato viene abbandonata la creazione del file .EXE per ritornare all'editor; ciò può essere compiuto anche premendo il tasto **Esc** in qualsiasi momento mentre è aperto il riquadro di comunicazione.

Si prema infine ancora una volta il tasto **Tab** per passare al pulsante più a destra, Guida: premendo Invio con questo pulsante evidenziato apparirà una guida all'uso di questo riquadro di comunicazione.

A questo punto si prema il tasto **Esc** per ritornare all'editor di QuickBASIC.

11.2 Creazione di un file .EXE che richiede BRUN45.EXE

Scegliendo la prima opzione di Genera: nel riquadro Crea un file EXE viene creato un file .EXE che richiede BRUN45.EXE.

```
(♦) Un file EXE che richiede BRUN45.EXE
```

Il file BRUN45.EXE deve essere reperibile al momento della chiamata dei file di questo tipo: contiene infatti del codice necessario ad implementare le funzioni del linguaggio. Si copi tale file su un disco appena formattato al quale si potranno aggiungere successivamente i file creati con questa opzione in modo da poterli eseguire senza richiedere altri dischi; una sola copia di BRUN45.EXE è in grado di servire tutti i file .EXE presenti sul disco.

I file .EXE che richiedono BRUN45.EXE presentano alcuni vantaggi.

- Sono molto più corti di quelli generati utilizzando l'opzione Crea un file EXE autonomo, per cui possono essere memorizzati molti più file eseguibili su un disco.
- Le variabili COMMON ed i file aperti sono preservati se si utilizza l'istruzione CHAIN, funzione che riveste notevole importanza nella creazione di sistemi di programmi che utilizzano dati condivisi (i programmi autonomi non preservano le variabili COMMON quando utilizzano le istruzioni CHAIN).

Il modulo di run-time di BRUN45.EXE risiede in memoria e non è necessario ricaricarlo per ogni programma facente parte di un sistema di programmi concatenati.

Si raccomanda di seguire i passi riportati di seguito qualora si intenda creare un file .EXE che richieda BRUN45.EXE.

- Si rendano disponibili i file necessari a QuickBASIC.
- Si carichai un programma QuickBASIC che sia stato eseguito con successo in precedenza e lo si salvi sul disco di libreria nell'unità B.
- Si selezioni Crea un file EXE nel menu Esegui dopo aver posto il disco di lavoro nell'unità A ed il disco di libreria nell'unità B.
- Si digiti il nuovo nome nel riquadro Nome del file EXE se si desidera cambiare il nome del file .EXE generato, altrimenti si lascino immutati i contenuti del riquadro e si prema il tasto **Tab** per passare all'area Genera codice Debug.
- Si digiti **D** marcando l'opzione Genera codice Debug e si indichi se si desidera generare il codice di debug con l'inclusione nel file eseguibile del codice per la generazione dei messaggi per il trattamento degli errori e l'indicazione della posizione degli errori al momento dell'esecuzione; si ricordi inoltre che questa opzione deve essere selezionata qualora si desideri la possibilità di interrompere l'esecuzione del programma con la combinazione di tasti **Ctrl+Break**. Questa opzione genera un codice più lungo ed in qualche misura più lento.
- Si prema nuovamente il tasto **Tab** per passare al primo elemento dell'area Genera.
- Si scelga Un file EXE che richiede BRUN45.EXE.
- Si prema quindi il tasto TAB per raggiungere il pulsante desiderato (Crea il file EXE o Crea il file EXE ed esci); il pulsante Crea il file EXE crea il file .EXE e ritorna all'editor di QuickBASIC, mentre il pulsante Crea il file EXE ed esci crea il file .EXE ed esce da QuickBASIC.

Una volta compiute tutte le scelte si prema **Invio**.

11.2.1 Il programma GRAFICI A BARRE

Si utilizzi il Programma 11.1, GRAFICI A BARRE, per creare il proprio primo file .EXE: tale programma è volutamente breve per permettere al lettore di concentrarsi sulla creazione del file eseguibile, inoltre le sue componenti possono essere facilmente modificate o cancellate e possono essere aggiunte nuove funzioni per personalizzarlo secondo le proprie necessità.

```
DECLARE SUB grafico (Entrata%())
DECLARE SUB etichetta (Mese$())
REM ** GRAFICI A BARRE **
```

```
' Capitolo 11
' Microsoft QuickBASIC 4.5  File: UQB1101.BAS

DIM Mese$(1 TO 12), Entrata%(1 TO 12)
CLS : DEFINT A-Z
' Legge i dati da mostrare nel grafico

FOR numero = 1 TO 12
   READ Mese$(numero), Entrata%(numero)
NEXT numero
 DATA Gen,84,  Feb,80,  Mar,74,  Apr,66
 DATA Mag,72,  Giu,50,  Lug,54,  Ago,58
 DATA Set,64,  Ott,72,  Nov,88,  Dic,86
' traccia il grafico e le etichette
CALL grafico(Entrata%())
CALL etichetta(Mese$())
COLOR 7
END

SUB grafico (Entrata%())
' Traccia i blocchi del grafico
   FOR numero = 1 TO 12
     COLOR numero
     blocchi = INT(Entrata%(numero) / 10)
     bloccoparz! = Entrata%(numero) / 10 - blocchi)
     LOCATE 16 - blocchi, 4 * numero + 12
     IF bloccoparz! > .3 AND bloccoparz! <= .7 THEN
       PRINT STRING$(2, 220);
     ELSEIF bloccoparz! > .7 THEN
       PRINT STRING$(2, 219);
     END IF
     FOR barra = blocchi - 1 TO 0 STEP -1
       LOCATE 16 - barra, 4 * numero + 12
       PRINT STRING$(2, 219);
     NEXT barra
   NEXT numero
END SUB

SUB etichetta (Mese$())
' Etichetta il grafico
   COLOR 11
   FOR riga = 1 TO 3
     FOR numero = 1 TO 12
       LOCATE riga + 17, 4 * numero + 12
       PRINT MID$(Mese$(numero), riga, 1);
     NEXT numero
   NEXT riga
   LOCATE 4, 32
   PRINT "Vendite per Mese";
```

```
  LOCATE 24, 1
END SUB
```

Programma 11.1 GRAFICI A BARRE

Il programma principale legge i dati delle vendite dell'ultimo anno di una società immaginaria e chiama due sottoprogrammi, il primo dei quali traccia sullo schermo delle barre che rappresentano le entrate di ciascun mese ed il secondo etichetta ciascuna barra con l'abbreviazione del nome del mese e scrive il titolo del grafico.

Il grafico viene tracciato in modo testo, utilizzando blocchi formati da due caratteri ASCII 219 e blocchi parziali formati da due caratteri ASCII 220; i grafici realizzati in modo testo non consentono una grande precisione, ma possono essere comunque utilizzati per mostrare i trend generali.

Il sottoprogramma che genera il grafico combina blocchi e blocchi parziali per creare la barra di ciascun mese.

```
blocchi = INT(Entrata%(numero) / 10)
bloccoparz! = Entrata%(numero) / 10 - blocchi)
```

I blocchi ed i blocchi parziali sono combinati dalla seguente porzione di programma:

```
LOCATE 16 - blocchi, 4 * numero + 12
IF bloccoparz! > .3 AND bloccoparz! <= .7 THEN
  PRINT STRING$(2, 220);                    'Scrive un blocco parziale
ELSEIF bloccoparz! > .7 THEN
  PRINT STRING$(2, 219);                    'Scrive un blocco completo
END IF
FOR barra = blocchi - 1 TO 0 STEP -1
  LOCATE 16 - barra, 4 * numero + 12
  PRINT STRING$(2, 219);                    'Scrive un blocco completo
NEXT barra
```

Le barre sono tracciate dall'alto in basso, sono larghe due caratteri e sono separate l'una dall'altra da due colonne vuote; la Tabella 11.2 mostra come sono calcolate le barre per due mesi. Ciascuna barra viene scritta con un colore differente variandolo ogni volta che si fa riferimento ad un numero di mese diverso.

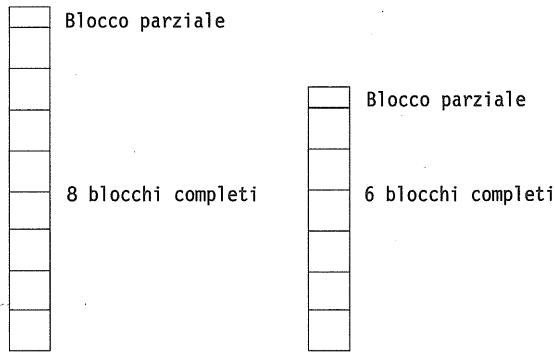


Tabella 11.2 Dai blocchi alla barre

Gennaio	Aprile
numero = 1	numero = 4
entrata%(1) = 84	entrata%(4) = 66
blocchi = INT(84/10) = 8	blocchi = INT(66/10) = 6
bloccoparz = 84/10 - 8 = 0.4	bloccoparz = 66/10 - 6 = 0.6
Traccia un blocco parziale poiché 0.4 è tra 0.3 e 0.7	Traccia un blocco parziale poiché 0.6 è tra 0.3 e 0.7
LOCATE del blocco parziale in 8,16 poiché 16 - 8 = 8 e 4 x 1 + 12 = 16	LOCATE del blocco parziale in 10,28 poiché 16 - 6 = 10 e 4 x 4 + 12 = 28
Calcola le posizioni dei blocchi completi:	Calcola le posizioni dei blocchi completi:
FOR barra = 7 TO 0 STEP -1	FOR barra = 5 TO 0 STEP -1
16 - 7, 16 = 9, 16	16 - 5, 28 = 11, 28
16 - 6, 16 = 10, 16	16 - 4, 28 = 12, 28
16 - 5, 16 = 11, 16	16 - 3, 28 = 13, 28
16 - 4, 16 = 12, 16	16 - 2, 28 = 14, 28
16 - 3, 16 = 13, 16	16 - 1, 28 = 15, 28
16 - 2, 16 = 14, 16	16 - 0, 28 = 16, 28
16 - 1, 16 = 15, 16	
16 - 0, 16 = 16, 16	

```
FOR numero = 1 TO 12
  COLOR numero
    .
    .
    .
```

Il sottoprogramma Etichetta utilizza un meccanismo di calcolo delle posizioni dello schermo simile per scrivere le abbreviazioni dei nomi dei mesi sotto le barre corrispondenti, quindi calcola la posizione del titolo del grafico e lo scrive. Si noti che il programma principale ripristina il colore originale prima di terminare: se ciò non fosse fatto tutte le scritte inviate allo schermo dopo l'esecuzione del programma sarebbero dell'ultimo colore specificato. La Figura 11.3 mostra l'output dell'esecuzione del Programma 11.1.

11.2.2 Creazione del file .EXE

Una volta digitato ed eseguito con successo il Programma 11.1, lo si salvi sul disco contenente i due file di libreria, BRUN45.LIB e BCOM45.LIB: a questo punto nell'unità

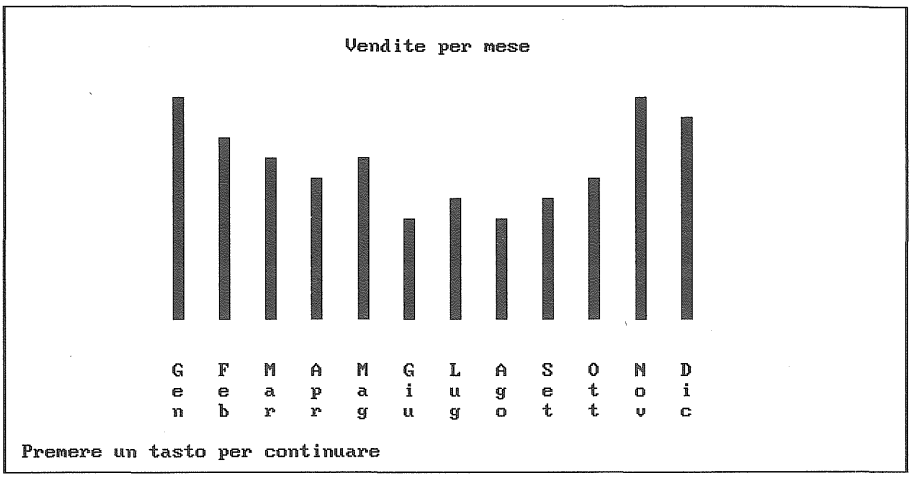


Figura 11.3 Output del Programma 11.1

B si troveranno tutti i file .LIB ed i file .BAS, come specificato nel riquadro Percorsi di ricerca del menu Opzioni presentato in precedenza. Si carichi il Programma 11.1 dall'unità B. Sono stati completati i primi due passi presentati precedentemente in questa sezione: restano da eseguire i passi da 3 ad 8, che possono essere riassunti come segue.

- Si selezioni Crea un file EXE nel menu Esegui.
- Non si modifichi il nome del file EXE.

```
Nome del file EXE        UQB1101.EXE
```

Si prema il tasto **Tab** per passare all'area Genera codice Debug.

- A questo punto si scelga se generare il codice di debug, tenendo conto del fatto che il file .EXE sarà più lungo nel caso si decida di attivare questa funzione premendo il tasto **D** (e facendo comparire una X tra le parentesi quadre).
- Si prema il tasto **Tab** per passare all'area Genera:.
- Si selezioni Un file EXE che richiede BRUN45.EXE.
- Si prema il tasto **Tab** per passare ai pulsanti in fondo al riquadro e si selezioni Crea il file EXE che riporta all'editor di QuickBASIC una volta generato il file eseguibile, oppure Crea il file EXE ed Esce, che esce da QuickBASIC una volta generato il file.

In questa dimostrazione non è stata attivata la generazione del codice di debug ed è stata selezionata Crea il file EXE nell'ultimo passo; nella Figura 11.4 vengono mostrati i messaggi emessi durante la creazione del file .EXE.

```
BC B:\UQB1101.BAS/T/C:512;
Compiler Microsoft (R) QuickBASIC Versione 4.50
(C) Copyright Microsoft Corporation 1982-1989
Tutti i diritti riservati.

44045 Byte disponibili
41882 Byte liberi

    0 Errori di avvertenza
    0 Errori gravi
LINK /EX UQB1101,A:\UQB1101.EXE,NUL,B:BRUN45.LIB

Microsoft (R) Overlay Linker  Versione 3.69
Copyright (C) Microsoft Corp 1983-1988.  Tutti i diritti riservati.
```

Figura 11.4 Output durante la creazione del file .EXE del Programma 11.1

Durante la compilazione QuickBASIC produce oltre al file .EXE (UQB1101.EXE) un file oggetto (UQB1101.OBJ); entrambi i file vengono memorizzati sul disco nell'unità A, seguendo le impostazioni compiute in precedenza nel riquadro di scelta dei percorsi. Una volta conclusa la compilazione la directory del disco nell'unità A contiene i seguenti file:

```
COMMAND   COM
QB        EXE
QB45QCK   HLP
BC        EXE
UQB1101   OBJ      <file oggetto
LINK      EXE
UQB1101   EXE      <file eseguibile
QB        INI
```

11.2.3 Esecuzione del file .EXE con BRUN45.EXE

I dettagli dell'esecuzione dei file .EXE dipendono dalla configurazione del sistema utilizzato; dal momento che il file eseguibile di questa dimostrazione è stato creato scegliendo la modalità che richiede BRUN45.EXE, questo file deve essere disponibile sul disco dei programmi (che, come riportato in precedenza dovrebbe essere formattato copiandovi i file

di sistema del DOS): si copi il file eseguibile appena creato in questo disco, che a questo punto dovrebbe contenere i tre seguenti file:

```
COMMAND.COM
BRUN45.EXE
UQB1101.EXE
```

Si inserisca il disco nell'unità A ed al prompt del DOS si digiti il nome del programma da eseguire.

```
A>UQB1101
```

Si prema **Invio** ed il programma sarà eseguito direttamente dal DOS: l'output sarà identico a quello mostrato nella Figura 11.3, con la differenza che non comparirà il prompt di QuickBASIC "Premere un tasto per continuare", che sarà sostituito dal prompt del DOS A> (se il programma .EXE fosse eseguito da un'unità differente, l'indicazione dell'unità contenuta nel prompt del DOS sarebbe ovviamente diversa).

Si esamini la directory dell'unità contenente il disco nel quale è memorizzato il file .EXE: contiene COMMAND.COM, BRUN45.EXE ed il file .EXE.

COMMAND.COM	Creato al momento della formattazione del dischetto con l'opzione di creazione dei file di sistema, occupa circa 23,6K di spazio su disco.
BRUN45.EXE	Modulo di run-time, occupa circa 77,4K di spazio.
UQB1101.EXE	File eseguibile, utilizza circa 4,6K di spazio.

Ogni volta che si aggiungerà un nuovo file eseguibile al disco dei programmi, lo spazio rimasto disponibile diminuirà solo della lunghezza del file eseguibile, dal momento che è richiesta una sola copia di BRUN45.EXE, che può essere utilizzato come modulo di run-time per tutti i file .EXE che lo richiedono.

11.3 Creazione di un file .EXE autonomo

I programmi autonomi vengono prodotti scegliendo l'opzione Un file EXE autonomo nel riquadro di creazione dei file .EXE e contengono tutte le routine di supporto reperibili in BRUN45.EXE, ragione per cui occupano più spazio sul disco rispetto ai file creati con l'opzione Un file EXE che richiede BRUN45.EXE.

I programmi autonomi non preservano le variabili elencate nelle istruzioni COMMON quando l'istruzione CHAIN trasferisce il controllo ad altri programmi, ma in compenso presentano alcuni vantaggi.

- Se il programma è semplice si può risparmiare lo spazio di RAM corrispondente alle routine del modulo di run-time non richieste.

- Non è necessaria la presenza del modulo di run-time su disco per l'esecuzione del programma, facilitando la scrittura di programmi che debbano essere copiati da terzi, che potrebbero non avere un modulo di run-time o non essere a conoscenza della sua necessità.

Per creare un file .EXE autonomo si compiano i passi riportati di seguito.

- Ci si accerti della disponibilità dei file di QuickBASIC necessari.
- Si carichi un programma QuickBASIC che sia stato eseguito con successo.
- Si richiami il comando Crea un file EXE del menu Esegui.
- Si digiti se richiesto un nuovo nome (privo di estensione) nel riquadro Nome del file EXE.
- Si prema il tasto **Tab** per passare all'area Genera codice Debug e si digiti **D** se si desidera la generazione del codice di debug.
- Si prema il tasto **Tab** per passare all'area Genera:.
- Si scelga l'opzione Un file EXE autonomo.
- Si selezioni l'opzione ritenuta più appropriata tra Crea il file EXE e Crea il file EXE ed esci: la prima riporta all'editor di QuickBASIC dopo avere creato il file eseguibile, mentre la seconda esce da QuickBASIC una volta generato il file.

Se durante l'esecuzione della procedura si decidesse di non creare il file .EXE si prema il tasto ESC per abbandonare l'operazione o si utilizzi il tasto **Tab** per spostarsi sul pulsante Abbandona e si prema **Invio**.

11.3.1 Il programma GRAFICI A TORTA

Per dimostrare la creazione di un file eseguibile autonomo si utilizzerà il Programma 11.2, GRAFICI A TORTA; dopo avere digitato il programma lo si mandi in esecuzione per verificarne il funzionamento: il programma principale crea numerosi vettori utilizzati dai sottoprogrammi e termina con un ciclo DO...LOOP che racchiude una chiamata al sottoprogramma Menu ed un blocco SELECT CASE utilizzato per richiamare gli altri sottoprogrammi.

```
DECLARE SUB Tracciamento ()
DECLARE SUB Informazioni ()
DECLARE SUB Menu (TastoPremuto$)
REM ** GRAFICI A TORTA **
' Capitolo 11
' Microsoft QuickBASIC 4.5  File: UQB1102.BAS

DIM percento!(0 TO 12), valore!(0 TO 12), angolo!(0 TO 12)
DIM col(0 TO 12) AS INTEGER, riga(0 TO 12) AS INTEGER
```

```
DEFINT A-Z

DO
   CALL Menu(TastoPremuto$)
   SELECT CASE TastoPremuto$
     CASE "I"
       CALL Informazioni
     CASE "T"
       CALL Tracciamento
     CASE "E"
       PRINT "CALL sottoprogramma Etichetta"
     CASE "U"
       EXIT DO
   END SELECT
LOOP
COLOR 7, 0: CLS

END

SUB Tracciamento
' Traccia il cerchio

   SHARED sec AS INTEGER, col AS INTEGER, riga AS INTEGER
   SHARED rad AS SINGLE, ecc AS SINGLE, col() AS INTEGER
   SHARED riga() AS INTEGER

   SCREEN 1, 0: CLS
   CIRCLE (col, riga), rad, 3, , , ecc!
   FOR settore = 1 TO sec
     LINE (col, riga)-(col(settore), riga(settore))
   NEXT settore
   colore = 1
   FOR settore = 1 TO sec
     IF settore = sec THEN
       ultimo = 1
       colore = 3
     ELSE ultimo = settore + 1
     END IF
     colpaint = (col(settore) + col(ultimo) - 2 * col) / 4
     rigapaint = (riga(settore) + riga(ultimo) - 2 * riga) / 4
     PAINT (col + colpaint, riga + rigapaint), colore, 3
     colore = colore + 1
     IF colore = 3 THEN colore = 1
   NEXT settore
   BEEP
   attesa$ = INPUT$(1)
   SCREEN 0: WIDTH 80: COLOR 7, 0

END SUB
SUB Informazioni
```

```
' Consente l'introduzione dei dati relativi al grafico
   SHARED sec AS INTEGER, col AS INTEGER, riga AS INTEGER, rad AS SINGLE
   SHARED ecc AS SINGLE, col() AS INTEGER, riga() AS INTEGER
   REDIM valore(0 TO 12) AS SINGLE, percento(0 TO 12) AS SINGLE
   REDIM angolo(0 TO 12) AS SINGLE
   CLS

   INPUT "Numero di settori"; sec
   INPUT "Centro del cerchio (col,riga)"; col, riga
   INPUT "Raggio del cerchio; rad!
   INPUT "Rapporto altezza/larghezza"; ecc!
   CLS : pi! = 3.14159
   FOR settore = 1 TO sec
     PRINT "Valore del settore "; settore;
     INPUT valore!(settore)
     valore!(0) = valore!(0) + valore!(settore)
   NEXT settore
   FOR settore = 1 TO sec
     angolo!(settore) = percento!(0) * pi! / 50
     percento!(settore) = valore!(settore) * 100 / valore!(0)
     percento!(0) = percento!(0) + percento!(settore)
     col(settore) = col + rad! * COS(angolo!(settore))
     riga(settore) = riga - rad! * SIN(angolo!(settore)) * ecc!
   NEXT settore
END SUB

SUB Menu (TastoPremuto$)
' Visualizza il menu delle selezioni
   COLOR 4, 0: CLS
   COLOR 9: LOCATE 5, 33: PRINT "MENU DEI GRAFICI A TORTA"
   LOCATE 10, 30: PRINT "("; : COLOR 4: PRINT "I";
   COLOR 9: PRINT ")nput informazioni"
   LOCATE 12, 30: PRINT "("; : COLOR 4: PRINT "T";
   COLOR 9: PRINT ")racciamento"
   LOCATE 14, 30: PRINT "("; : COLOR 4: PRINT "E";
   COLOR 9: PRINT ")tichette"
   LOCATE 16, 30: PRINT "("; : COLOR 4: PRINT "U";
   COLOR 9: PRINT ")scita"
   COLOR 27, 1
   LOCATE 22, 24: PRINT CHR$(219);
   COLOR 10: PRINT "DIGITARE UNA LETTERA (I, T, E od U) ";
   COLOR 27: PRINT CHR$(219);
   COLOR 7, 0
   DO
     BEEP: TastoPremuto$ = UCASE$(INPUT$(1))
   LOOP WHILE INSTR("ITEU", TastoPremuto$) = 0
END SUB
```

Programma 11.2 GRAFICI A TORTA

Eseguendo il programma viene visualizzato sullo schermo il menu mostrato nella Figura 11.5 nel quale la prima lettera di ciascuna selezione è racchiusa tra parentesi ed evidenziata con un colore differente da quello del resto della scritta in modo da differenziare dal resto della scritta la lettera da digitare.

```
MENU DEI GRAFICI A TORTA

(I)nput informazioni
(T)racciamento
(E)tichette
(U)scita

Digitare una lettera (I, T, E od U)
```

Figura 11.5 Menu dei grafici a torta

Il sottoprogramma Informazioni richiede l'introduzione del numero di settori di cui si compone la torta, la posizione del centro del cerchio, la lunghezza del raggio e la sua eccentricità (rapporto tra la altezza e larghezza), il cui valore di default è 1, che nella maggioranza degli schermi video produce un'ellisse più alta che larga (di norma per ottenere un cerchio rotondo occorre un valore inferiore di 1).

Il sottoprogramma richiede inoltre le informazioni da utilizzare nel calcolo delle dimensioni di ciascun settore, somma i valori ricevuti dall'utente e trova la percentuale rispetto al totale di ciascun elemento; successivamente vengono applicate alcune formule trigonometriche per calcolare le posizioni dei punti del cerchio dove passano le suddivisioni dei settori. La Figura 11.6 mostra a titolo di esempio i valori immessi per un grafico con cinque settori.

```
Numero di settori ? 5
Centro del cerchio (col,riga)? 160,100
Raggio del cerchio ? 70
Rapporto altezza/larghezza ? .85_
```

Figura 11.6 Informazioni immesse per il grafico

```
Valore del settore 1 ? 82
Valore del settore 2 ? 45
Valore del settore 3 ? 28
Valore del settore 4 ? 38
Valore del settore 5 ? 62_
```

Figura 11.7 Valori immessi per i cinque settori

Il seguente ciclo FOR...NEXT calcola il totale dei valori immessi dall'utente memorizzando le somme parziali in valore!(0):

```
FOR settore = 1 TO sec
  PRINT "Valore del settore "; settore;
  INPUT valore!(settore)
  valore!(0) = valore!(0) + valore!(settore)
NEXT settore
```

I valori immessi nei vari passi del ciclo e le corrispondenti somme parziali per i dati dell'esempio sono:

Settore%	Valore!(settore%)	Valore!(0)
1	82	82
2	45	127
3	28	155
4	38	193
5	62	255

Un secondo ciclo FOR...NEXT calcola le percentuali dei vari settori rispetto al totale, l'angolo di rotazione di ciascun settore e le coordinate degli estremi del suo arco.

```
FOR settore = 1 TO sec
  angolo!(settore) = percento!(0) * pi! / 50
  percento!(settore) = valore!(settore) * 100 / valore!(0)
  percento!(0) = percento!(0) + percento!(settore)
  col(settore) = col + rad! * COS(angolo!(settore))
  riga(settore) = riga - rad! * SIN(angolo!(settore)) * ecc!
NEXT settore
```

Ci si soffermi innanzitutto sul calcolo della percentuale dei singoli settori e sulle somme parziali delle percentuali.

```
  percento!(settore) = valore!(settore) * 100 / valore!(0)
  percento!(0) = percento!(0) + percento!(settore)
```

Il valore di valore!(0), ovvero la somma dei valori dell'esempio, è 255, per cui le percentuali di ciascun settore e le somme parziali sono:

Settore	Valore! (settore)	Totale prima	Percento! (settore)	Totale dopo
1	82	0	32.16	32.16
2	45	32.16	17.64	49.80
3	28	49.80	10.98	60.78
4	38	60.78	14.90	75.69
5	62	75.69	24.31	100

Un cerchio completo comprende 2 * pi radianti (dove pi rappresenta la costante pi greco), per cui il valore in radianti dell'angolo corrispondente ad una particolare percentuale è:

```
percento * 2 * pi! / 100 cioe' percento * pi! / 50
```

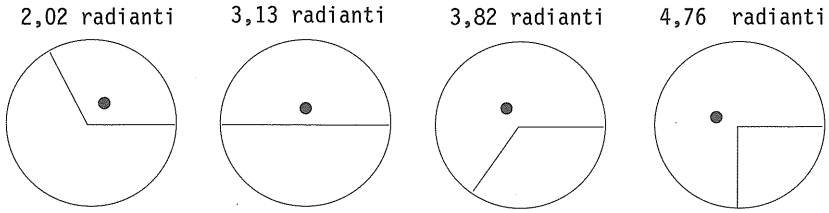
e l'angolo iniziale di un settore è:

```
angolo!(settore) = percento!(0) * pi! / 50
```

dove percento!(0) è la somma parziale calcolata prima di aggiungere la percentuale del settore corrente; i risultati dei calcoli eseguiti servendosi delle percentuali date sono:

Settore	Percento!(0) * pi! / 50	Angolo in radianti
1	0 * 3.14159 / 50	0
2	32.16 * 3.14159 / 50	2.02
3	49.80 * 3.14159 / 50	3.12
4	60.78 * 3.14159 / 50	3.82
5	75.69 * 3.14159 / 50	4.76

Gli angoli vengono utilizzati per calcolare gli estremi di ciascun settore sulla circonferenza.



I calcoli possono essere resi più efficienti dividendo 3.14159 per 50 prima di iniziare il ciclo ed utilizzando il risultato nelle moltiplicazioni di ciascun percento!(0) all'interno di questo, effettuando, nel caso dell'esempio, una divisione invece di cinque.

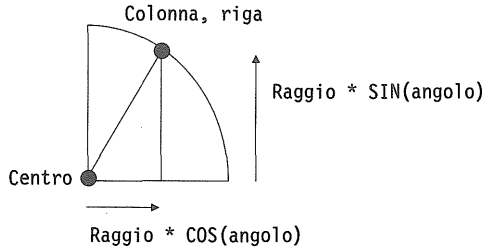
Una volta trovati gli angoli, i loro valori sono utilizzati per calcolare le coordinate degli estremi di ciascun settore sulla circonferenza.

```
col(settore) = col + rad! * COS(angolo!(settore))
riga(settore) = riga - rad! * SIN(angolo!(settore)) * ecc!
```

Per ogni estremo le relazioni matematiche tra coordinate ed angolo sono:

```
colonna = raggio * COS(angolo)
riga = raggio * SIN(angolo)
```

Le espressioni possono essere dimostrate come segue.



Nei calcoli è necessario negare il segno dell'espressione utilizzata per calcolare la riga per tenere conto del fatto che le coordinate delle righe dello schermo sono misurate dall'alto verso il basso.

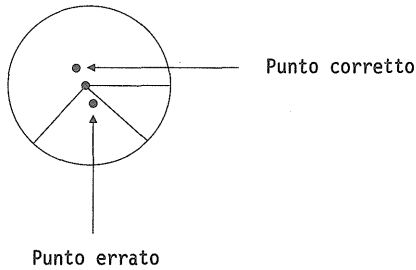
```
riga = -raggio * SIN(angolo)
```

Per calcolare le coordinate delle righe viene anche utilizzata l'eccentricità, ovvero il rapporto tra altezza e larghezza del cerchio, il cui valore immesso dall'utente nel sottoprogramma Informazioni è assegnato alla variabile ecc! e viene utilizzato per tracciare un cerchio vero (cioè che appaia rotondo sullo schermo video). Utilizzando le equazioni date, le coordinate dell'esempio sono:

Settore	Angolo	COS	SIN	Coordinate
1	0	1.0	0	230.100
2	2.02	-0.435	.401	130.46
3	3.13	-1.0	.0123	90.99
4	3.82	-0.779	-0.627	105.137
5	4.76	.0431	-0.999	163.159

Utilizzando queste coordinate, a cui vanno aggiunte quelle del centro del cerchio ricevute in precedenza, i punti sono collocati sulla circonferenza come nella figura seguente.

Il sottoprogramma Tracciamento utilizza una nuova istruzione: SHARED. Questa istruzione consente al sottoprogramma la condivisione di variabili con il programma principale ed



altre procedure del modulo senza che sia necessario passarle come parametri; le istruzioni SHARED utilizzate nel sottoprogramma sono:

```
SHARED sec AS INTEGER, col AS INTEGER, riga AS INTEGER
SHARED rad AS SINGLE, ecc AS SINGLE, col() AS INTEGER
SHARED riga() AS INTEGER
```

Il cerchio viene tracciato dall'istruzione:

```
CIRCLE  (col, riga), rad, 3, , , ecc!
```

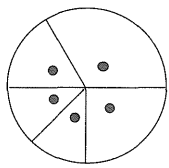
^ centro, ^ raggio, ^ colore, ^ eccentricità

Le linee di separazione dei settori della torta sono tracciate da:

```
LINE (col, riga) - (col(settore), riga(settore))
```

^dal centro del cerchio ^alla circonferenza

I valori dei punti sulla circonferenza - col(settore) e riga(settore) - sono quelli calcolati nel sottoprogramma Informazioni e sono condivisi dai vari sottoprogrammi grazie all'istruzione SHARED; il tracciamento delle linee produce i seguenti settori:



Una volta divisa la torta in settori viene utilizzata l'istruzione PAINT per colorare l'interno di ciascun settore.

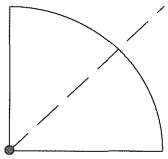
```
PAINT (col + colpaint, riga + rigapaint), colore, 3
```

^colora a partire da questo punto, ^colore dello sfondo, ^fino a raggiungere questo punto

Occorre compiere alcune considerazioni al riguardo del calcolo delle coordinate del punto dal quale iniziare la colorazione dei settori: tale punto deve infatti giacere all'interno del

settore corrispondente e la colorazione procede da esso in tutte le direzioni fino a raggiungere il colore di confine; per calcolare la posizione di tale punto vengono eseguiti i passi che seguono.

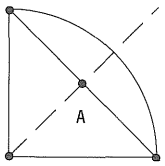
- Si immagini una retta passante per il centro del cerchio che divida il settore in parti uguali.



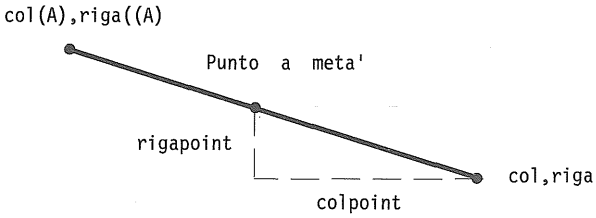
- Si consideri poi il punto che giace all'intersezione di questa retta con la corda avente per estremi gli estremi del settore sulla circonferenza

```
col(A) = (col(settore) + col(ultimo)) / 2
riga(A) = (riga(settore) + riga(ultimo)) / 2
```

Ad esempio:



- Il punto del passo 2 sembra soddisfacente ma è troppo vicino alla circonferenza: sarebbe meglio scegliere un punto più vicino al centro, ma non troppo, ad esempio il punto a metà tra quello del passo 2 ed il centro del cerchio.



La distanza tra il centro del cerchio ed il punto da cui iniziare la colorazione può essere calcolata (facendo uso di alcune manipolazioni algebriche) dall'istruzione QuickBASIC che segue.

```
colpaint = (col(settore) + col(ultimo) - 2 * col) / 4
rigapaint = (riga(settore) - riga(ultimo) - 2 * col) / 4
```

Che nell'esempio producono i seguenti risultati:

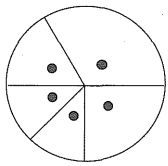
Settore	Col(settore)	Col(ultimo)	Colpaint
1	230	130	10
2	130	90	-25
3	90	105	-31.25
4	105	163	-13
5	163	230	18.25

Settore	Riga(settore)	Riga(ultimo)	Rigapaint
1	100	46	-13.5
2	46	99	-13.75
3	99	137	9
4	137	159	24
5	159	100	14.75

Le coordinate dei punti iniziali di colorazione dei settori si ottengono sommando i valori di rigapaint e colpaint alle coordinate del centro del cerchio (160, 100).

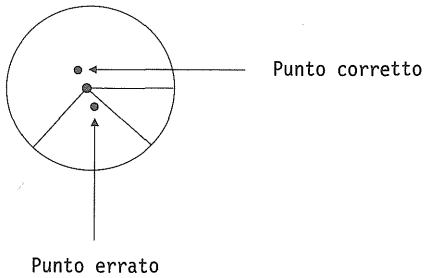
Settore	Col(settore)	Riga(settore)
1	170	86.5
2	90	72.5
3	77.5	118
4	114	148
5	176.5	129.5

Ecco la disposizione dei punti da cui inizia la colorazione:



Questo metodo di calcolo è talmente semplificato da non funzionare correttamente per settori che occupino più del 50% del cerchio: si consideri a titolo di esempio una torta comprendente tre settori occupanti rispettivamente il 60%, 25% e 15%: il punto di inizio della colorazione calcolato dal Programma 11.2 per il settore da 60% cade fuori dal settore stesso.

Per ottenere il punto corretto i valori delle variabili colpaint e rigapaint devono in questo caso essere sottratti dalle coordinate del centro del cerchio.




```
PAINT (col - colpaint, riga - rigapaint), colore, 3
```

Si potrebbe utilizzare un blocco IF...THEN...ELSE per prevedere anche il caso dei settori che occupano più del 50% della torta, assegnando in tal caso valori negativi alle variabili colpaint e rigapaint.

Esiste un'altra possibilità: che cosa succederebbe se un settore occupasse esattamente il 50% della torta? La soluzione di questo problema di programmazione viene lasciata al lettore.

La variabile colore cambia ad ogni settore: tenendo conto del fatto che nel modo grafico utilizzato (screen 1) sono disponibili solamente quattro colori, i colori si ripeteranno se esistono più di quattro settori. Una volta tracciata la torta il programma emette una nota ed attende la pressione di un tasto per consentire all'utente di studiare il grafico ed annotare eventuali modifiche dei dati; premendo un tasto qualsiasi si ritorna al menu. La Figura 11.8 mostra il grafico generato dai dati della Figura 11.7.

Il lettore può scrivere il proprio sottoprogramma di visualizzazione delle etichette in modo da abbellire come meglio preferisce il grafico. Per aggiungere testo alle schermate grafiche

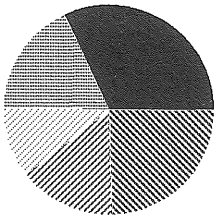


Figura 11.8 Grafico a torta

in modo screen 1 si tenga presente che le lettere sono delle stesse dimensioni di quelle dei modi testo a 40 colonne e possono essere posizionate ovunque nello schermo utilizzando l'istruzione LOCATE.
Una volta che il Programma 11.2 funzioni correttamente, lo si salvi con il nome UQB1102.BAS.

11.3.2 Creazione del file autonomo

Un file .EXE autonomo viene creato esattamente come un file eseguibile che richiede BRUN45.EXE, con una eccezione: la scelta compiuta nell'area Genera:.

Si apra il menu Esegui e si scelga il comando Crea un file EXE: nel riquadro Nome del file EXE comparirà il nome del file .EXE da creare.

```
Nome del file EXE   UQB1102.EXE
```

Se si desidera un nome diverso lo si digiti nel riquadro, che altrimenti deve essere lasciato immutato.

Si prema il tasto **Tab** per passare all'opzione Genera codice Debug: se si desidera la generazione del codice di debug descritto in precedenza si prema il tasto **D**. Si utilizzi il tasto **Tab** per passare quindi all'area Genera: e si scelga Un file EXE autonomo premendo **Freccia Giù**. Si selezioni il pulsante Crea il file EXE o Crea il file EXE ed esce a proprio piacimento: se si intende proseguire la sessione QuickBASIC una volta generato il file .EXE, si scelga il primo pulsante, mentre se si preferisce tornare al DOS si scelga il secondo; in entrambi i casi si prema **Invio** per creare il file eseguibile. La schermata prodotta durante la creazione del file è mostrata nella Figura 11.9.

```
BC B:\UQB1102.BAS/T/C:512;
Compiler Microsoft (R) QuickBASIC Versione 4.50
(C) Copyright Microsoft Corporation 1982-1989

Tutti i diritti riservati.
44045 Byte disponibili
39704 Byte liberi

    0 Errori di avvertenza
    0 Errori gravi
LINK /EX UQB1102,A:\UQB1102.EXE,NUL,B:BCOM45.LIB

Microsoft (R) Overlay Linker  Versione 3.69
Copyright (C) Microsoft Corp 1983-1988.  Tutti i diritti riservati.
```

Figura 11.9 Output durante la creazione del file .EXE del Programma 11.2

11.3.3 Esecuzione del file autonomo

I file .EXE autonomi vengono avviati dal DOS nello stesso modo di quelli che richiedono BRUN45.EXE, con la differenza che non è necessaria la presenza di tale file: i risultati sono identici a quelli dell'esecuzione dello programma corrispondente in ambiente QuickBASIC, salvo che il prompt a fine esecuzione sarà diverso. I file .EXE autonomi comprendono le routine di supporto contenute in BRUN45.EXE, per cui questo tipo di file necessita più spazio su disco dei corrispondenti .EXE che richiedono BRUN45.EXE: esaminando la directory del proprio disco dei programmi si potrà constatare che il file .EXE del Programma 11.2 (UQB1102.EXE) è molto più lungo di UQB1101.EXE; di seguito viene compiuto un raffronto tra i file prodotti dai due metodi.

.EXE autonomo	EXE che richiedono BRUN45.EXE
UQB1102.EXE 45.9K	UQB1101.EXE 4.6K
	BRUN45.EXE 77.4K

Ad ogni creazione di un nuovo file .EXE autonomo verrà occupato molto più spazio sul disco, a differenza del caso dei file che richiedono BRUN45.EXE: questo, pur essendo un file lungo, viene utilizzato in una sola copia da tutti gli eseguibili dello stesso disco.

Come prima menzionato, si possono utilizzare altre configurazioni di dischi, ad esempio il file BRUN45.EXE potrebbe essere salvato nel disco di lavoro invece che nel disco dei programmi, purché sia reso disponibile al momento dell'esecuzione dei file .EXE.

11.4 Sommario

In questo capitolo sono stati discussi due metodi di creazione dei file eseguibili direttamente dalla linea di comando dell'MS- DOS; la creazione di tali file coinvolge numerosi file di QuickBASIC in più rispetto all'esecuzione dei programmi .BAS nell'ambiente integrato, per cui si rende necessaria la creazione di dischi di lavoro aggiuntivi, sui quali si dovranno ripartire i file di supporto basandosi sulla configurazione dei dischi del proprio sistema.

Per essere eseguiti i file eseguibili possono richiedere o no la presenza di un file speciale chiamato BRUN45.EXE, che contiene le routine necessarie ad implementare le funzioni del linguaggio: la scelta se avvalersi o no di tale file dipende da varie considerazioni che comprendono tra l'altro l'occupazione dello spazio sul disco e la realizzazione di sistemi di file concatenati. Il primo programma di questo capitolo traccia dei grafici a barre sullo schermo in modo testo, mentre il secondo traccia un grafico a torta servendosi delle istruzioni grafiche di QuickBASIC.

Capitolo 12

Librerie Quick

12.1 Vantaggi delle librerie Quick
12.2 Creazione di una libreria Quick
12.3 Libreria Quick UTIL1
12.4 Aggiunte ad una libreria Quick
12.5 Sommario

Il capitolo finale tratta una delle caratteristiche più efficaci di QuickBASIC, di cui ci si può servire per realizzare degli strumenti di programmazione con tutti i programmi di utilità che sono stati creati o che lo saranno. QuickBASIC mette a disposizione due strumenti per creare due diversi tipi di libreria: queste librerie indipendenti vengono create con il Microsoft Library Manager, che è il file con nome LIB.EXE.

Un tipo di libreria viene identificato dalla estensione .LIB ed il secondo tipo di libreria è identificato dall'estensione .QLB: queste librerie speciali, dette librerie Quick, permettono di aggiungere ad un qualunque programma QuickBASIC procedure frequentemente utilizzate e possono contenere procedure scritte in altri linguaggi Microsoft, quali ad esempio il Microsoft C. Quando QuickBASIC crea una libreria Quick, crea simultaneamente anche una libreria indipendente contenente le stesse procedure in una forma differente: si ottengono coì due librerie invece che una.

È possibile creare entrambe i tipi di libreria sia dall'ambiente di programmazione che dalla linea di comando del DOS. In questo capitolo verranno trattate solamente le librerie Quick.

12.1 Vantaggi delle librerie Quick

Si pensi alla libreria Quick come ad un insieme di una o più procedure che può essere unito al QuickBASIC quando la libreria viene caricata con QuickBASIC. Mentre si sviluppa un programma lungo è possibile aggiungere alla libreria Quick dei moduli che si sono

completati ed eseguiti con successo. I file sorgente dei moduli completati possono essere messi da parte sino a che non si desideri migliorarli od aggiornarli: è quindi possibile caricare la libreria Quick con QuickBASIC ed ogni programma che si crea o carica avrà accesso istantaneo a tutte le procedure nella libreria.

Le procedure della libreria Quick si comportano come le istruzioni QuickBASIC: ad esempio possono essere chiamate da un programma od essere eseguite dalla finestra immediata rendendo quindi possibile la verifica degli effetti di una procedura della libreria Quick prima di farne uso in altri programmi.

È possibile includere nelle librerie Quick routine scritte in altri linguaggi e tutte, quando si carica la libreria Quick che le contiene insieme con QuickBASIC, si comporteranno come una estensione del linguaggio QuickBASIC. Se si sviluppano programmi insieme ad altri programmatori, le librerie Quick facilitano la realizzazione e l'aggiornamento di un insieme di procedure comuni.

Si potrebbe offrire una libreria di procedure originali per la distribuzione commerciale e tutti i programmatori QuickBASIC sarebbero in grado di servirsene immediatamente per migliorare il loro lavoro.

12.2 Creazione di una libreria Quick

Quando si crea una libreria Quick nuova, questa contiene automaticamente tutti i moduli presenti nell'ambiente QuickBASIC ed ogni altra libreria Quick che sia stata caricata quando si è inizializzato QuickBASIC. Se si carica un programma e si desidera porre nella libreria solamente alcuni moduli è necessario eliminare esplicitamente quelli non desiderati: questa operazione può essere eseguita con il comando Chiudi file del menu File.

Per controllare quali moduli sono caricati, è possibile prendere visione dell'elenco del comando SUB del menu visualizza, ma questo elenco non mostra quali procedure contiene una libreria caricata; è possibile elencare le procedure di una libreria con il programma di utilità QLBDUMP.BAS. Una libreria Quick deve essere autocontenuta, cioè tutte le procedure chiamate da una procedura in una libreria Quick devono essere in essa contenute. Esistono almeno due fattori da considerare nel creare una libreria Quick: in primo luogo quali file siano necessari e, per secondo, il metodo da utilizzare per crearla.

12.2.1 File necessari

Prima di creare una libreria Quick ci si assicuri di disporre dei file necessari. Se non ci si serve di un disco fisso potrebbe essere necessario più di un dischetto.

Quando QuickBASIC, quando non riesce a trovare un file, richiede un percorso; quando ciò accade si inserisca il dischetto corretto e si risponda alla richiesta.

La Tabella 12.1 elenca e descrive brevemente i file che dovrebbero essere accessibili a QuickBASIC; tutti questi file, che sono inclusi nel pacchetto QuickBASIC, trovano facilmente spazio su un dischetto da 3"1/2 e quindi, se si dispone di un'unità a dischetti da 5"1/4, si dovrebbero creare uno o più dischetti di lavoro contenenti i file nella tabella.

Tabella 12.1 File per la libreria Quick

Nome del file	Uso
QB.EXE	Dirige la creazione di una libreria Quick
BC.EXE	Crea il file oggetto dal codice sorgente
LINK.EXE	Collega i file oggetto
LIB.EXE	Ottiene le librerie indipendenti dai moduli oggetto
BLQ45.LIB	Fornisce le routine necessarie alla libreria Quick

12.2.2 Libreria Quick creata da QuickBASIC

Quando si crea una libreria Quick questa potrebbe essere una libreria completamente nuova oppure dell'aggiornamento di una libreria esistente.

Se si sta aggiornando una libreria si dovrebbe inizializzare QuickBASIC con l'opzione /L fornendo il nome della libreria che deve essere aggiornata come argomento della linea di comando, come illustrato nel seguito del capitolo.

- Si acceda al menu Esegui e si selezioni il comando Crea libreria, come illustrato nella Figura 12.1: quando si preme **Invio**, si vedrà il riquadro di comunicazione illustrato nella Figura 12.2.
- Si digiti il nome della libreria che si desidera creare nella casella Nome del file libreria Quick: se si introduce solamente un nome base, QuickBASIC aggiungerà automaticamente l'estensione .QLB quando crea la libreria; è possibile introdurre un qualsiasi nome base ed una qualsiasi estensione purché si osservino le regole del DOS sui nomi di file e se non si desidera una estensione si termini il nome base con un punto. Seguono alcuni esempi.
  - **Nome di file senza estensione** Si digiti il nome makefile nella casella Nome del file libreria Quick senza punto e senza estensione: non si prema ancora **Invio**, dato che si devono ancora operare altre scelte di Crea libreria.

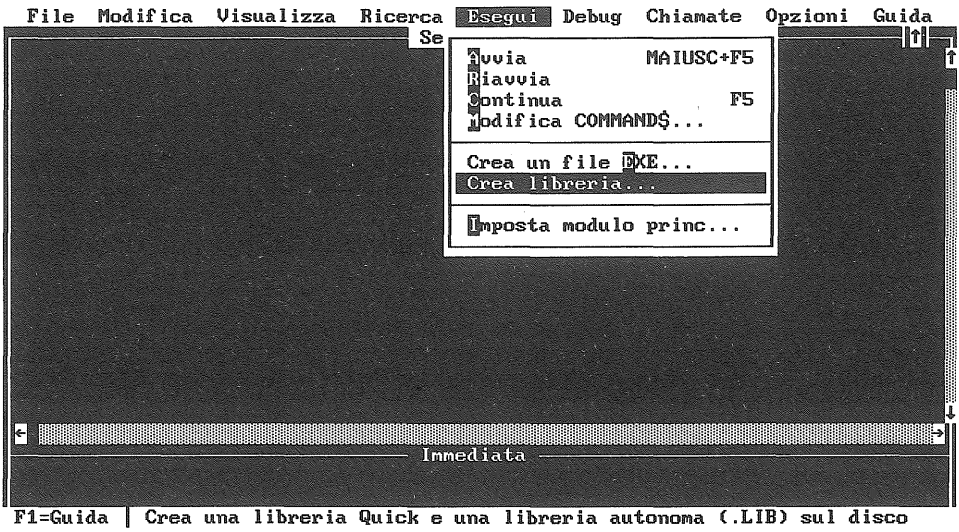


Figura 12.1 Comando Crea libreria selezionato

```
Nome del File libreria Quick: MAKEFILE
```

QuickBASIC assegnerà al file nome MAKEFILE.QLB

- **Nome di file con estensione** Si digiti il nome di file seguito da un punto ed una estensione di tre lettere: non si prema ancora **Invio**, dato chevi sono ancora altre scelte di Crea libreria da operare.

```
Nome del File libreria Quick: MAKEFILE.QLB
```

QuickBASIC assegnerà al file il nome MAKEFILE.QLB; è possibile assegnare una qualunque estensione valida di tre lettere conforme con le estensioni di file del DOS; se si utilizza una estensione diversa da ".QLB", sarà necessario ricordarsi l'estensione e farne uso ogni volta che si carica la libreria Quick.

- **Nome di file terminante con un punto, ma senza estensione** Si digiti il nome del file seguito da un punto: non si prema ancora **Invio**, dato che si devono ancora operare altre scelte di Crea libreria.

```
Nome del File libreria Quick: MAKEFILE.
```

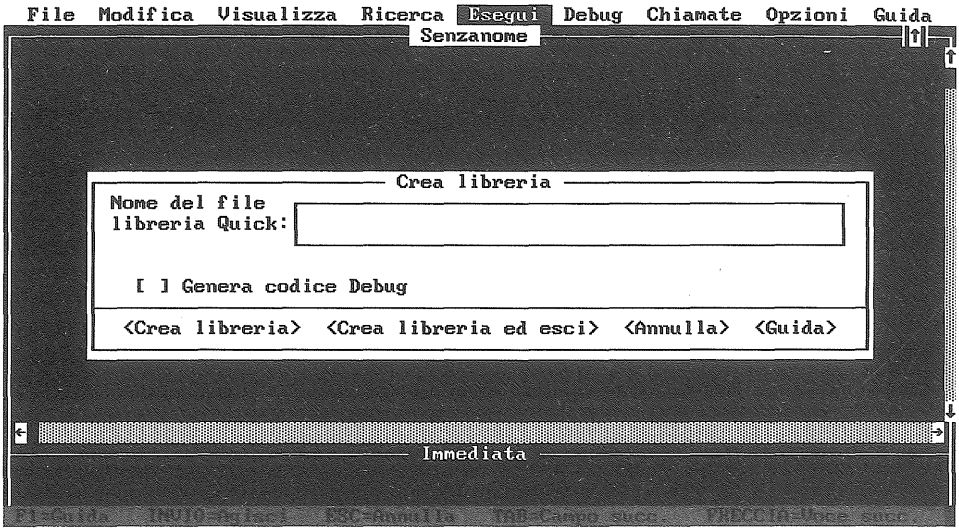


Figura 12.2 Richiesta del nome per la libreria Quick

Il file viene creato senza estensione; se il nome di file della libreria Quick non ha estensione è necessario specificare il nome nello stesso modo (MAKEFILE.) ogni volta che la si carica; se non si pospone il punto, QuickBASIC cercherà invano il nome di file con nome base (MAKEFILE) ed estensione .QLB.

- Si prema il tasto **Tab** ed il cursore si sposta alla casella di controllo Genera codice Debug: non si selezioni questa opzione ora; se lo si fosse fatto la libreria sarebbe più grande ed il programma verrebbe eseguito più lentamente. Questa opzione fornisce solamente una piccola quantità di controllo dell'errore, principalmente riguardante il controllo dei limiti degli array.
- Si deve ora operare la scelta finale: la libreria Quick viene creata quando si attiva uno dei due pulsanti Crea libreria che sono visualizzati nella parte inferiore dello schermo nella Figura 12.2, mentre il terzo pulsante annulla l'operazione.
  - **Crea libreria** Consente di rimanere nell'ambiente QuickBASIC dopo che si è creata la libreria Quick; se è questa la scelta desiderata, si prema il tasto **Invio** quando è evidenziato il pulsante.
  - **Crea libreria ed esci** Se si desidera creare la libreria ed uscire al DOS si prema il tasto **Tab** in modo ad evidenziare il pulsante di comando Crea libreria ed esci: si tornerà al DOS dopo che la libreria sarà stata creata.
  - **Annulla** Se si cambia idea e si desidera annullare l'operazione si prema il tasto **Tab** sino ad evidenziare il pulsante di comando Annulla, quindi si prema il tasto **Invio**.

12.2.3 Dimostrazione della creazione di una libreria Quick

Si potrebbe disporre di un qualunque numero di piccoli frammenti di programma o di moduli che vengono utilizzati in molti programmi; anziché introdurre ognuno di questi da tastiera ogniqualvolta si scrive un nuovo programma è possibile introdurli una sola volta ed inserirli in una libreria Quick comune. Sarà quindi possibile caricare la libreria dalla linea di comando quando si desidera includere i moduli in un nuovo programma.

Questa sezione si serve dei passi elencati nella sezione precedente per illustrare come creare una libreria Quick di brevi utilità. Si esegua QuickBASIC e si digiti il Programma 12.1, Sottoprogramma di stampa della data ed ora.

```
DECLARE SUB Orologio ()
REM ** Sottoprogramma di stampa di data ed ora **
' Capitolo 12
' Microsoft QuickBASIC 4.5  File: UQB1201.BAS
' Da porre in una Quick Library; attivato da: CALL Orologio
' Usare il menu Modifica per vedere il sottoprogramma Orologio

SUB Orologio
' Scrive data ed ora sul dispositivo 1
  PRINT #1, "Data "; DATE$
  PRINT #1, "Ore  "; TIME$
END SUB
```

Programma 12.1 Sottoprogramma di stampa della data e dell'ora

Si osservi che questo programma si serve di un'istruzione PRINT #1 per inviare la data e l'ora ad una qualunque periferica di I/O che si sia precedentemente specificata in un'istruzione OPEN. Le periferiche di I/O disponibili sono elencate nella Tabella 12.2.

Tabella 12.2 Periferiche accettate per l'I/O

Nome	Periferica	Modo I/O accettato
COM1:	Porta seriale #1	Input ed Output
COM2:	Porta seriale #2	Input ed Output
CONS:	Schermo	Solamente Output
KYBD:	Tastiera	Solamente Input
LPT1:	Stampante #1	Solamente Output
LPT2:	Stampante #2	Solamente Output
LPT31:	Stampante #3	Solamente Output
SCRN:	Schermo	Solamente Output

Le due periferiche di I/O più comuni sono uno schermo di visualizzazione ed una stampante e le istruzioni OPEN per queste sono:

```
OPEN "SCRN:" FOR OUTPUT AS #1   'output allo schermo
OPEN "LPT1:" FOR OUTPUT AS #1   'output alla stampante
```

Questa trattazione assume che si stia utilizzando un sistema a due unità con QuickBASIC nell'unità A ed il dischetto dei file dati nell'unità B; se si utilizza un sistema a disco fisso, si sostituisca l'appropriata designazione dell'unità per il sistema in uso.

Il Programma 12.1 è molto breve e non è un programma eseguibile da solo, ma, quando lo si include in una libreria, può essere richiamato da qualunque programma che faccia uso di quella libreria. Dopo aver introdotto il programma lo si memorizzi su disco con il nome UQB1201.BAS.

Si potrebbe trovare inutile creare una libreria per un solo piccolo programma dato che se fosse tutta la libreria consistesse solo in quello sarebbe inutile, ma in questo capitolo si aggiungeranno alla libreria altre utilità: questo programma illustra solamente come cominciare con una piccola libreria ed espanderla in seguito.

- Si acceda al menu Esegui e si selezioni il comando Crea libreria.
- Si digiti il nome della libreria che deve essere creata, ma non si prema ancora **Invio**.

  ```
  Nome del file Libreria Quick: b:UTIL1.QLB
  ```

- Si prema il tasto Tab due volte in modo da passare oltre il comando Genera codice Debug: non si desidera selezionare questa opzione.
- Si prema il tasto **Tab** per spostare il cursore dal pulsante evidenziato Crea libreria a quello di comando Crea libreria ed esci e quindi si prema **Invio**.

QuickBASIC procede alla creazione della libreria: il programma visualizza sullo schermo una sequenza di messaggi mentre crea la libreria; visualizza anche qualunque errore venga rilevato. Nell'esempio illustrato nella Figura 12.3 non è stato rilevato alcun errore, la libreria Quick UTIL1.QLB è stata creata e QuickBASIC ha restituito il controllo al DOS, come era stato richiesto.

Ora che il computer è tornato al DOS si controllino le directory delle unità A e B (supponendo che i file QuickBASIC.EXE e LIB.EXE siano nel'unità A ed i file sorgente nell'unità B): nella Figura 12.4 sono illustrate alcune directory tipiche.

```
BC B:\UQB1201.BAS/T/C:512;

Compiler Microsoft (R) QuickBASIC Versione 4.50

(C) Copyright Microsoft Corporation 1982-1989.

Tutti i diritti riservati.


44061 Byte disponibili

43275 Byte liberi
```

```
    0 Errori di avvertenza
    0 Errori gravi
LINK /QU UQB1201,b:\UTIL1\QLB,NUL,A:\BQLB45.LIB;

Microsoft (R) Overlay Linker  versione 3.69
Copyright (C) Microsoft Corp. 1983-1988.  Tutti i diritti riservati.

LIB B:\UTIL1.LIB+UQB1201;

Microsoft (R) Library Manager  versione 3.14
Copyright (C) Microsoft Corp. 1983-1988.  Tutti i diritti riservati.

A>
```

Figura 12.3 Schermata finale di Crea libreria

```
Directory di B:\
UQB1201      BAS        372
UTIL1        QLB       5591
UTIL1        LIB       1551

Directory di A:\
QB           EXE     278804
BC           EXE      97481
BQLB45       LIB      24789
LINK         EXE      69133
LIB          EXE      65643
QB           INI         48
UQB1201      OBJ        886
```

Figura 12.4 Directory dopo la creazione di UTIL1

Si osservi che UTIL1.QLB e UTIL1.LIB sono stati registrati sul disco nell'unità B: si era specificato questo percorso quando si è dato il nome alla libreria Quick (b:UTIL1). Si osservi inoltre che il file oggetto (UQB1201.OBJ) è stato registrato sul disco nell'unità A.

Quando QuickBASIC crea una libreria Quick, dirige il lavoro dei programmi BC.EXE, LINK.EXE e LIB.EXE, poi combina quello che producono in due librerie: una libreria Quick (.QLB) ed una libreria indipendente (.LIB). Quando il processo è completato esiste un file oggetto (.OBJ) per ognuno dei moduli nel programma. I messaggi illustrati nella Figura 12.3 delineano i passi per la creazione di una libreria.

- Le prime tre linee della Figura 12.3 illustrano che QB.EXE chiama il compilatore QuickBASIC BC per elaborare il file QuickBASIC UQB1201.BAS.

  Il compilatore crea un file oggetto (.OBJ) che è dapprima utilizzato per creare il file di libreria Quick (UTIL1.QLB); quando BC.EXE ha completato il proprio compito elenca il numero di byte disponibili ed il numero di byte liberi; il messaggio finale per questa parte dell'elaborazione non riporta errori di avvertenza e gravi.

- In prossimità della parte centrale dell'elenco di messaggio si osserva la parola LINK: questa indica che il file LINK.EXE sta operando; i file da collegare, UQB1201, UTIL1.QLB, NUL e BLQB45.LIB sono visualizzati sulla stessa linea. La voce NUL nell'elenco indica che non è stato creato un file per il listato; se si fosse desiderato ottenere tale file si sarebbe dovuto fornire il nome di una periferica in questo punto dell'elenco; l'ultimo file dell'elenco, BQLB45.LIB, fornisce routine necessarie alla libreria Quick. Quando tutti questi file sono stati correttamente collegati, il Library Manager termina.

- Una delle linee finali sullo schermo inizia con la parola LIB; indicando che il programma LIB.EXE riceve il controllo e crea una libreria indipendente (UTIL1.LIB) dal modulo oggetto. Il programma dimostrativo consiste di un singolo modulo, quindi LIB.EXE crea un file oggetto (UQB1201.OBJ); le librerie generate sono UTIL1.QLB (la libreria Quick) e UTIL1.LIB (la libreria indipendente).

I file con l'estensione .OBJ non sono più necessari e potrebbero essere eliminati, ma per il momento si lasci il file UQB1201.OBJ in modo da poter osservare gli altri file oggetto con l'aggiunta di moduli alla libreria. I file con estensione .LIB hanno grande importanza e dovrebbero essere conservati; queste librerie parallele vengono utilizzate per creare i file eseguibili dei programmi. Se si sta sviluppando del software professionale volto ad altri programmatori QuickBASIC, si dovrebbero consegnare al cliente sia la libreria Quick sia la libreria indipendente; senza le librerie .LIB i clienti non potranno infatti servirsi delle routine di libreria in file eseguibili generati da QuickBASIC.

12.3 Libreria Quick UTIL1

Per assicurarsi che la libreria Quick UTIL1 funzioni, si carichino QuickBASIC e la libreria dalla linea di comando con:

```
AQB /L B:UTIL1
```

dove /L indica al computer di caricare una libreria, B: è il percorso in cui verrà trovata la libreria e UTIL1 è il nome della libreria. Compare uno schermo quasi completamente vuoto e, se ci si serve del menu Visualizza per controllare le SUB, non si troverà nulla: benché la libreria sia caricata, non si osserverà alcun elenco delle procedure nella libreria.

Si potrebbe scrivere un breve programma per provare la libreria UTIL1, ma è molto più semplice servirsi della finestra Immediata per introdurre un comando CALL al sottoprogramma Orologio in UTIL1.

Si prema il tasto **F6** in modo da spostare il cursore nella finestra Immediata e si predisponga la periferica I/O necessaria; se si desidera utilizzare come Output lo schermo, si seguano i seguenti passi:

- si digiti **cls** e si prema **Invio**;
- si prema nuovamente **Invio** per tornare alla finestra Immediata quando compare il messaggio "Premere un tasto per continuare";
- si digiti **open "scrn:" for output as #1** e si prema **Invio**;
- si digiti **call orologio** e si prema **Invio**.

La data e l'ora vengono determinate da Orologio ed inviate allo schermo.

Se si desidera inviare la data e l'ora alla stampante, ci si assicuri che questa sia on line e quindi, se si era predisposta la periferica #1 come schermo, si seguano i seguenti passi:

- si digiti **close #1** e si prema **Invio**;
- si predisponga la stampante come periferica di I/O #1;
- si digiti **open "lpt1:" for output as #1** e si prema **Invio**;
- si digiti **call orologio** e si prema **Invio**.

Le informazioni su data ed ora vengono inviate alla stampante.

12.4 Aggiunte ad un libreria Quick

Se si desidera scrivere dei programmi di aspetto professionale è spesso necessario servirsi dei tasti funzione e cursore; quando questi tasti vengono premuti restituiscono un codice di due caratteri ASCII: il primo carattere è sempre il codice nullo, CHR$(0), il secondo è nel campo dei codici ASCII di un singolo carattere. Ad esempio, il tasto **Freccia Destra** genera il codice nullo seguito dal codice 59.

Una procedura che identifichi quale tasto è stato premuto sarebbe una comoda utilità: il Programma 12.2, Sottoprogramma di scansione della tastiera, consiste di alcuni commenti e di un sottoprogramma (ScansioneTasti) per assolvere a tale funzione.

```
DECLARE SUB ScansioneTasti (ta$)
REM ** SUB di scansione della tastiera **
' Capitolo 12
' Microsoft QuickBASIC 4.5  File: UQB1202.BAS
' Da porre in una Quick Library; attivato da: CALL ScansioneTasti(ta$)
' Usare il menu Modifica per vedere il sottoprogramma ScansioneTasti(ta$)

SUB ScansioneTasti (ta$)
' Attende la pressione di un tasto e riporta il suo codice esteso
   ta$ = ""
   DO WHILE ta$ = ""
     ta$ = INKEY$
   LOOP
   IF LEN(ta$) = 2 THEN
     ta$ = "0 +" + STR$(ASC(RIGHT$(ta$, 1)))
   ELSE
     ta$ = STR$(ASC(ta$))
   END IF
END SUB
```

Programma 12.2 Sottoprogramma di scansione della tastiera

Quando si preme un tasto, il programma ScansioneTasti lo legge con la funzione INKEY$ ed assegna alla variabile ta$ ciò che ha letto; la funzione INKEY$ restituisce una stringa di un carattere per i tasti codificati ASCII (come A, a, 1, 2 e così via), mentre restituisce una stringa di due caratteri per i codici ASCII estesi (come F1, F2 e gli spostamenti del cursore). Il primo carattere di una stringa di due caratteri è sempre 0 (carattere nullo), il secondo è un codice ASCII (utilizzato per altri tasti nel set ASCII); nella manualistica Microsoft è reperibile elenco completo di questi codici estesi.

Quando si preme un tasto, il Programma 12.2 assegna dapprima il valore del tasto alla variabile ta$, poi determina la lunghezza della variabile ta$: se la lunghezza è 2, ta$ viene modificata dal'istruzione:

```
ta$ = "0 +" + STR$(ASC(RIGHT$(ta$,1)))
```

Questa istruzione concatena la parte di codice nullo ("0+") alla stringa equivalente al codice ASCII del secondo carattere del codice esteso.

Se la lunghezza di ta$ è 1 la variabile viene modificata in modo da essere la stringa equivalente del codice ASCII restituito permettendo il riconoscimento dei caratteri non stampabili (come il tasto **Invio**).

Quando si compie un'aggiunta alla libreria Quick si deve attribuire alla libreria risultante un nome differente: nell'esempio seguente verrà attribuito il nome UTIL2. Per dare inizio al processo di creazione di una nuova libreria si carichino QuickBASIC e UTIL1 dalla linea di comando del DOS.

```
AQB /L B:UTIL1
```

supponendo che l'unità B contenga i programmi e le librerie. Si prema **Invio** e quando compare la schermata di modifica di QuickBASIC si esegua uno dei seguenti passi:

- si richiami il Programma 12.2 se lo si è già introdotto e memorizzato;
- si digitia ora il Programma 12.2.

Quando il Programma 12.2 è visualizzato nella finestra, si acceda al comando Crea libreria del menu Esegui, si prema **Invio**, si digiti nella casella Nome del file libreria Quick **b:util2** come nuovo nome della libreria e quindi si prema **Tab** in modo da spostare il cursore al pulsante di comando Crea libreria ed esci: lo schermo dovrebbe presentarsi come nella Figura 12.5. Si prema **Invio** per creare la nuova libreria.

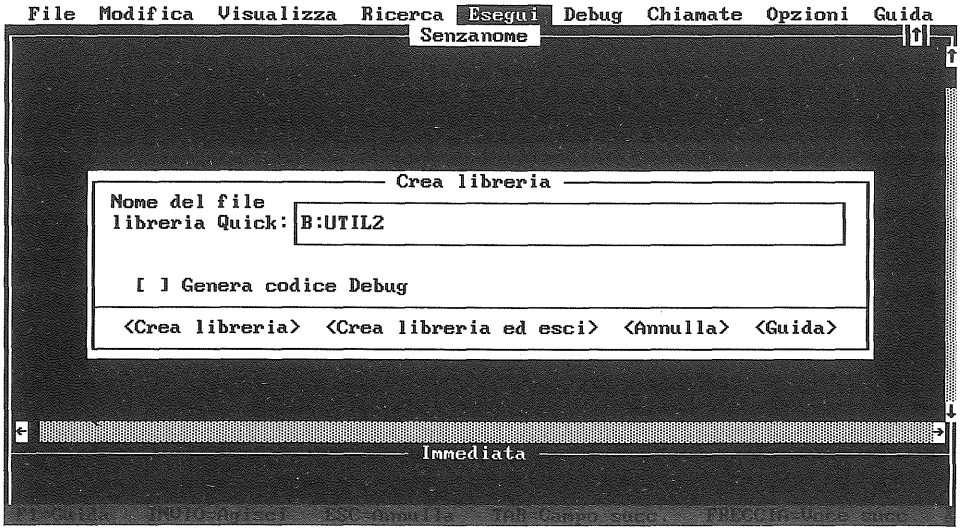


Figura 12.5 Nome del file libreria Quick introdotto

Ancora una volta è possibile osservare sullo schermo l'evoluzione nella creazione della libreria: il risultato è illustrato nella Figura 12.6.

```
BC B:\UQB1202.BAS/T/C:512;
Compiler Microsoft (R) QuickBASIC Versione 4.50
(C) Copyright Microsoft Corporation 1982-1989.
Tutti i diritti riservati.

44061 Byte disponibili
43601 Byte liberi

    0 Errori di avvertenza
```

```
    0 Errori gravi
LINK /QU /NOE /NOD UQB1202+B:\UTIL1.LIB,
                   B:\UTIL2.QLB,NUL,A:\BQLB45.LIB;

Microsoft (R) Overlay Linker  versione 3.69
Copyright (C) Microsoft Corp. 1983-1988.  Tutti i diritti riservati.

LIB B:\UTIL2+UQB1202+B:\UTIL1.LIB;

Microsoft (R) Library Manager  versione 3.14
Copyright (C) Microsoft Corp. 1983-1988.  Tutti i diritti riservati.

A>_
```

Figura 12.6 Creazione della libreria Quick UTIL2

Si è creata con successo una libreria Quick (UTIL2) e si è tornati al DOS; è possibile verificare ciò controllando le directory delle unità A e B.

La dimostrazione ha aggiunto file in entrambe le unità, come è illustrato nella Figura 12.7 seguente.

```
Directory di B:\
UQB1201     BAS     372
UTIL1       QLB    5591
UTIL1       LIB    1551
UQB1202     BAS     491
UTIL2       QLB    6206
UTIL2       LIB    2581

Directory di A:\
QB          EXE    278804
BC          EXE    97481
BQLB45      LIB    24789
LINK        EXE    69133
LIB         EXE    35643
UQB1201     OBJ      886
```

```
UQB1202      OBJ      1064
```

Figura 12.7 Directory dopo che UTIL2 è stata creata

Si osservi che vi sono due file .OBJ nella directory dell'unità A, uno per ognuno dei moduli: poiché non sarà più necessario farne uso è possible se si desidera eliminare questi file oggetto, ma ci si ricordi di lasciare i file .LIB e .QLB nell'unità B.

12.4.1 Prova della libreria Quick UTIL2

Si carichino QuickBASIC e la libreria Quick UTIL2 dal DOS con il seguente comando

```
AQB /L B:UTIL2
```

Prima di provare UTIL2 si acceda al menu Visualizza, si selezioni SUB e si provi a cercare i sottoprogrammi; si ricordi che i sottoprogrammi sono ora nella libreria Quick UTIL2 e, poiché una libreria Quick è essenzialmente un file binario, non è possibile vederne il contenuto da QuickBASIC o con un qualunque altro editor di testi. Uno dei dischi di distribuzione QuickBASIC contiene tuttavia in una sottodirectory chiamata SOURCE un file con il nome QLBDUMP.BAS, che permette di elencare tutte le procedure in un libreria specificata.

Si compiano i seguenti passi per vedere i contenuti di una libreria Quick:

- si carichi QuickBASIC;
- si richiami e si esegua QLBDUMP.BAS;
- si digiti il nome della libreria Quick che si desidera esaminare; in questo caso si introduca il nome come segue:

  ```
  B:UTIL2.QLB
  ```

Se il file specificato esiste e si tratta di una libreria Quick, il programma QLBDUMP visualizza un elenco di nomi simbolici contenuti nella libreria; nell'elenco di nomi simbolici sono inclusi i nomi delle procedure nella libreria. Ad esempio, per la libreria dimostrativa QLBDUMP.BAS genera l'elenco illustrato nella Figura 12.8; si osservino le ultime voci elencate sotto Code Symbols: compaiono i nomi SCANSIONETASTI e OROLOGIO che sono i nomi delle procedure contenute.

```
Code Symbols

_brtcl
_execve
_exit
```

```
_main
_spawnve
_TEXT_END
SCANSIONETASTI
SetUEvent
OROLOGIO

Data Symbols

FIWRQQ
FJSRQQ
b_erradr
b_ULVars
b_errlin
b_errnum
FISRQQ
_end
STKHQQ
_errno
_edata
FIARQQ
FJARQQ
FICRQQ
FIJRQQ
FIDRQQ
FIERQQ
b_errmod
b_ULSymSeg
_environ
```

Figura 12.8 Output di QLBDUMP.BAS

Si provi ora UTIL2; questa libreria contiene due sottoprogrammi: Orologio e Scansione-Tasti(ta$). Si era provato Orologio in UTIL1, ma ci si assicuri che sia presente in UTIL2 premendo F6 in modo da spostare il cursore nella finestra Immediata e ci si assicuri anche

che la stampante sia on line, se si intende utilizzarla come periferica di output. Si compiano quindi i passi che seguono:

- si digiti **cls** e si prema **Invio**;
- si prema nuovamente **Invio** per tornare alla finestra Immediata;
- si introduca la corretta istruzione OPEN per la periferica di output (scrn: o lpt1:);
- si digiti **call orologio** e si prema **Invio**.

La data e l'ora vengono inviate alla periferica di output (schermo o stampante) ed il sottoprogramma nella libreria Quick dimostrativa ha stampato:

```
Data 01-01-1990
Ore  12:14:13
```

Per provare il sottoprogramma ScansioneTasti, si compiano i seguenti passi:

- si digiti **call scansionetasti$** e si prema **Invio**;
- si prema il tasto **Freccia Su** e quindi si prema **Invio** per tornare alla finestra Immediata;
- si digiti **print ta$** e si prema **Invio**.

La stampante o lo schermo dovrebbero stampare:

```
0+72
```

il codice carattere esteso per il tasto **Freccia Su** e nella parte inferiore dello schermo viene visualizzato il messaggio "Premere un tasto per continuare".

Ora si effettui un'altra prova compiendo i seguenti passi:

- si prema un tasto qualunque per tornare alla finestra Immediata;
- si sposti il cursore verso l'alto fino alla linea con il comando CALL ScansioneTasti(ta$) e si prema nuovamente **Invio**;
- si prema il tasto **F4** e quindi Invio per tornare alla finestra Immediata;
- si digiti **print ta$** nella finestra immediata e si prema **Invio**.

La stampante dovrebbe stampare:

```
0 + 62
```

il codice carattere esteso per il tasto **F4**, proprio sotto al precedente codice per il tasto **Freccia Su**, provando che UTIL2, la libreria Quick dimostrativa, ha passato il test.

12.4.2 Ulteriori aggiunte

Il Programma 12.3, SUB Ritardo, è l'ultima componente da aggiungere alla libreria Quick: genera un ritardo di tempo di un numero specificato di secondi. La lunghezza del ritardo deve essere impostata nel programma principale chiamando poi la routine di ritardo e la funzione di QuickBASIC TIMER termina il ritardo quando sono trascorsi i secondi specificati. Si introduca il Programma 12.3 e lo si memorizzi come UQB1203.BAS, si esca quindi da QuickBASIC e si torni al DOS. Si carichino QuickBASIC, UQB1203.BAS e la libreria Quick UTIL2 dalla linea di comando con il seguente comando:

```
AQB B:UQB1203 /L B:UTIL2
```

Segue il listato.

```
DECLARE SUB Ritardo (sec!)
REM ** Ritardo SUB **
' Capitolo 12
' Microsoft QuickBASIC 4.5  File: UQB1203.BAS
' Da porre in una Quick Library; attivato da: CALL Ritardo(sec!)
' Il programma principale deve impostare i secondi (sec!)
' Usare il menu Modifica per vedere il sottoprogramma Ritardo(sec!)

SUB Ritardo (sec!)
' Introduce un ritardo in secondi specificato dal programma principale
  Riparte! = TIMER
  DO UNTIL TIMER > = Riparte! + sec!
  LOOP
END SUB
```

Programma 12.3 Subroutine Ritardo

Quando si sono caricati tutti questi file, il Programma 12.3 compare nella finestra di modifica; si seguano gli stessi passi già visti per creare la libreria Quick: si acceda al menu Esegui, si selezioni il comando Crea libreria, si introduca il nome del file libreria Quick UTIL3, si prema il tasto **Tab** in modo da evidenziare il pulsante di comando Crea libreria ed esci e si prema **Invio**.

Si osservi lo schermo mentre viene creata la libreria Quick ed a questo punto si potrà riconoscere quando operano per la creazione della libreria Quick UTIL3 il compilatore, il linker ed il library manager. Quando l'operazione è terminata, si osservi come la directory del disco nell'unità B sia cresciuta; la directory dell'unità B è illustrata nella Figura 12.9.

```
Directory di B:\

UQB1201    BAS     372
UTIL1      QLB    5591
UTIL1      LIB    1551
UQB1202    BAS     491
```

```
UTIL2      QLB     6206
UTIL2      LIB     2581
UQB1203    BAS      441
UTIL3      QLB     6385
UTIL3      LIB     6385
```

Figura 12.9 Directory del disco dimostrativo

Poiché UTIL3.QLB contiene tutti i sottoprogrammi di UTIL1.QLB e di UTIL2.QLB, si conservi UTIL3.QLB e si eliminino le altre due; si esegua la stessa operazione per i file .LIB: si cancellino i file UTIL1.LIB ed UTIL2.LIB, ma si conservi UTIL3.LIB. È inoltre possibile eliminare dal disco nell'unità A tutti i file .OBJ genereati durante la creazione delle librerie Quick; la Figura 12.10 illustra i file rimasti su entrambe i dischi dopo che sono stati eliminati il file .OBJ.

```
Directory di A:\
QB         EXE    278804
BC         EXE     97481
BQLB45     LIB     24789
LINK       EXE     69133
LIB        EXE     35643
QB         INI        48

Directory di B:\
UQB1201    BAS       372
UQB1202    BAS       491
UQB1203    BAS       441
UTIL3      QLB      6385
UTIL3      LIB      6385
```

Figura 12.10 Directory dopo le eliminazioni

12.4.3 Prova della libreria Quick UTIL3

Per provare le prime due librerie ci si è serviti della finestra Immediata: ora è possibile utilizzare il Programma 12.4, Prova UTIL3, per provare tutti e tre i sottoprogrammi nella

libreria Quick UTIL3. Non si digiti ancora il Programma 12.4, si carichino dapprima QuickBASIC ed UTIL3 con il seguente comando:

```
AQB /L b:UTIL3
```

Segue il listato.

```
REM ** Prova UTIL3 **
' Capitolo 12
' Microsoft QuickBASIC 4.5  File: UQB1204.BAS
' Utilizzato per provare la Quick Library UTIL3

CLS

DO
   INPUT "Output a schermo o stampante (S o P)"; ta$
LOOP WHILE INSTR("SsPp", ta$) = 0
SELECT CASE ta$
   CASE "S", "s"
     OPEN "SCRN:" FOR OUTPUT AS #1
   CASE "P", "p"
     OPEN "LPT1:" FOR OUTPUT AS #1
END SELECT

CALL Orologio
sec! = 5
LOCATE 12, 25: PRINT "Attendere prego, sto pensando...."
CALL Ritardo(sec!)
LOCATE 12, 25: PRINT SPACE$(27)
PRINT "Premere un tasto per stampare i codici estesi."
PRINT : PRINT "Premere Invio per uscire."

REM ** Attende la pressione di un tasto **
DO
   CALL ScansioneTasti(ta$)
   PRINT #1, : PRINT #1, ta$

LOOP UNTIL ta$ = " 13"
CLOSE #1
END
```

Programma 12.4 Prova UTIL3

Non appena avvenuta l'esecuzione si presenterà uno schermo quasi completamente vuoto con il cursore nella finestra di modifica: si digiti ora il Programma 12.4. Si memorizzi il Programma 12.4 con il nome UQB1204.BAS, se si intende farne uso, ci si assicuri che la stampante sia on line e si esegua il programma. Dapprima compare nella parte superiore dello schermo la richiesta "Output a schermo o stampante (S o P)?". Si digiti S o P; se si digita P, la data e l'ora vengono inviate alla stampante, mentre se si digita S vengono inviate allo schermo.

Al centro dello schermo viene quindi stampato il seguente messaggio:

```
Attendere prego, sto pensando...
```

Il computer chiama il sottoprogramma Ritardo ed attende per cinque secondi (poiché il ritardo di tempo era stato impostato a questa lunghezza) e quindi si è invitati a premere un tasto.

```
Premere un tasto per stampare i codici estesi.
Premere Invio per uscire.
```

Quando si preme un tasto la stampante o lo schermo stampano il codice carattere del tasto. La Figura 12.11 illustra la data e l'ora che sono state stampate ed i codici carattere per i tasti premuti: **Freccia Su**, **Freccia Giù**, **Freccia Sinistra**, **F1**, **F2**, **F3**, **F4** ed **Invio**. Tutti i tre sottoprogrammi nella libreria Quick UTIL3 hanno funzionato correttamente.

```
Data 01-02-1990
Ore  17:39:03
0+72
0+80
0+75
0+77
0+59
0+60
0+61
0+62
 13

Premere un tasto per continuare
```

Figura 12.11 Output del Programma 12.4

Questa dimostrazione della libreria Quick inizia appena a descrivere le utilità che potrebbero essere poste in una libreria Quick; è possibile utilizzare UTIL3 come base ed aggiungere un insieme più completo di utilità, oppure disfarsene e costruire una propria libreria Quick. Probabilmente si desidererà infatti creare alcune librerie Quick con diversi scopi: con un po' di pianificazione le librerie Quick risparmieranno molto tempo quando si scrive, carica ed esegue un programma.

12.5 Sommario

Le librerie Quick permettono di aggiungere le procedure frequentemente utilizzate ad un qualunque programma QuickBASIC; le procedure della libreria Quick si comportano come istruzioni QuickBASIC, possono essere chiamate da un programma QuickBASIC od eseguite dalla finestra Immediata.

Una volta disponibili i file appropriati è possibile creare librerie Quick dall'ambiente QuickBASIC ed aggiungere procedure ad una libreria Quick precedentemente creata in modo da formarne una più completa.

Questo capitolo contiene gli esempi della creazione di una libreria Quick nuova e dell'aggiunta di procedure ad un libreria esistente.

L'argomento conclude l'introduzione all'uso di QuickBASIC, versione 4.5. Quando si tratta un ambiente di programmazione con capacità tanto vaste risulta impossibile includere tutti gli aspetti del sistema.

Appendice A

Parole riservate

Le seguenti parole sono riservate in ambiente QuickBASIC: non devono essere quindi utilizzate come etichette, nomi di variabili o procedure.

ABS
ACCESS
ALIAS
AND
ANY
APPEND
APP
AS
ASC
ATN
BASE
BEEP
BINARY
BLOAD
BSAVE
BYVAL
CALL
CALLS
CASE
CDBL
CDECL
CHAIN
CHDIR
CHR$
CINT
CIRCLE
CLEAR
CLNG
CLOSE
CLS
COLOR
COM
COMMAND$
COMMON
CONST
COS
CSNG
CSRLIN
CVD
CVDMBF
CVI
CVS
CVSMBF
DATA
DATE$
DECLARE
DEF
DEFDBL
DEFINT
DEFLNG
DEFSNG
DEFSTR
DIM
DO
DOUBLE
DRAW
ELSE

ELSEIF
END
ENDIF
ENVIRON
ENVIRON$
EOF
EQV
ERASE
ERDEV
ERDEV$
ERL
ERR
ERROR
EXIT
EXP
FIELD
FILEATTR
FILES
FIX
FOR
FRE
FREEFILE
FUNCTION
GET
GOSUB
GOTO
HEX$
IF
IMP
INKEY$
INP
INPUT
INPUT$
INSTR
INT
INTEGER
IOCTL
IOCTL$
IS
KEY
KILL
LBOUND
LCASE$
LEFT$
LEN
LET
LINE
LIST
LOC
LOCAL
LOCATE
LOCK
LOF
LOG
LOOP
LPOS
LPRINT
LSET
LTRIM$
MID$
MKD$
MKDIR
MKDMBF$
MKI$
MKL$
MKS$
MOD
NAME
NEXT
NOT
OCT$
OFF
ON
OPEN
OPTION
OR
OUT
OUTPUT
PAINT
PALETTTE
PCOPY
PEEK
PEN
PLAY
PMAP
POINT
POKE
POS
PRESET
PRINT

PSET
PUT
RANDOM
RANDOMIZE
READ
REDIM
REM
RESET
RESTORE
RESUME
RETURN
RIGHT$
RMDIR
RND
RSET
RTRIM$
RUN
SADD
SCREEN
SEEK
SEG
SELECT
SETMEM
SGN
SHARED
SHELL
SIGNAL
SIN
SINGLE
SLEEP
SOUND
SPACE$
SPC
SQR
STATIC
STEP
STICK
STOP
STR$
STRIG
STRING
STRING$
SUB
SWAP
SYSTEM
TAB
TAN
THEN
TIME$
TIMER
TO
TROFF
TRON
TYPE
UBOUND
UCASE$
UEVENT
UNLOCK
UNTIL
USING
VAL
VARPTR
VARPTR$
VARSEG
VIEW
WAIT
WEND
WHILE
WIDHT
WINDOW
WRITE
XOR

La McGraw-Hill pubblica in tutto il mondo centinaia di libri di informatica per lo studio, la professione e il tempo libero. La produzione in lingua italiana comprende:

MICROCALCOLATORI

88 386 0049-X Murray W.H., Pappas C.H., *L'Assembler per l'80286/80386*
88 386 0009-0 Peckham H., Ellis Jr W., Lodi E., *Il BASIC e il Commodore 64 in pratica*
88 386 0008-2 Peckham H., *Il BASIC e il PC-IBM in pratica*
88 386 0030-9 Ettlin W., Solberg G., *Il BASIC Microsoft*
88 386 0150-X Jones E., *Come usare dBASE IV*
88 386 0040-6 Barnes L., *Come usare il dBASE III*
88 386 0056-2 Jones E., *Come usare il dBASE III PLUS*
88 386 0026-0 Ettlin W., *Come usare il Multiplan*
88 386 0019-8 Baras E.M., *Come usare il Symphony*
88 386 0064-3 Ettlin W., *Come usare il WordStar 4.0*
88 386 0095-3 Matthews M.S., Matthews C., *Come usare Lotus 1-2-3 Release 3 Versione inglese*
88 386 0054-6 Brownstein M., *Come usare Microsoft Word*
88 386 0149-6 Perry G.M., *Come usare Microsoft Word 5*
88 386 0080-5 Cowart R., *Come usare Microsoft Works*
88 386 0072-4 Cobb S., *Come usare Reflex*
88 386 0148-8 Matthews M.S., Matthews B.C., *Come usare Wordstar 5.5.*
88 386 0050-3 Andersen D., Cooper C., Dempsey B., *dBASE III in pratica*
88 386 0093-7 Carrabis J.D., *dBASE III PLUS Applicazioni in rete locale*
88 386 0039-2 Hixson A.C., *Excel Guida avanzata*
88 386 0046-5 Jones E., *Excel per personal MS-DOS*
88 386 0152-6 Comi A., *Elaborazioni grafiche con il personal computer*
88 386 0055-4 Heilborn J., *Guida al Commodore 128*
88 386 0001-5 Heilborn J., Talbott R., *Guida al Commodore 64*
88 386 0043-0 Byers T.J., *Guida al PC AT IBM*
88 386 0021-X Graham L.J., Field T., *Guida al PC-IBM*
88 386 0076-7 Wood S., *Guida al Turbo Pascal 4.0*
88 386 0044-9 Kater D.A., Kater R.L., *Guida alle stampanti Epson*
88 386 0138-0 Etlin W., Solberg G., *Il GW-BASIC per il PC1 Olivetti Prodest*
88 386 0041-4 Ettlin W., Solberg G., *Il GW-BASIC per Personal Computer Olivetti*
88 386 0126-7 Biow L., *Introduzione a Quattro*
88 386 0085-6 Baeseler F., Heck B., *Introduzione al Desktop Publishing*
88 386 0037-6 Kruglinski D., *Introduzione al Framework*
88 386 0042-2 Watt D., *Il Logo per il Commodore 64*
88 386 0053-8 Kyd C.W., *Lotus 1-2-3: applicazioni finanziarie*
88 386 0164-X Williams A.T., *Lotus 1-2-3 Release 2.2 Versione inglese*

88 386 0025-2	Morgan C., Waite M., *Il manuale 8086/8088*
88 386 0058-9	Cramer W., Kane G., *Il manuale MC 68000*
88 386 0137-2	Siechert C., Wood C., *Il manuale MS-DOS 3.20 per il PC1 Olivetti Prodest*
88 386 0034-1	Hoffman P., *Il manuale MSX*
88 386 0047-3	Siechert C., Wood C., *Il manuale MS-DOS 3.20*
88 386 0159-3	Siechert C., Wood C., *Il manuale MS-DOS 3.30*
88 386 0158-5	Siechert C., Wood C., *Il manuale MS-DOS 4*
88 386 0052-X	Hoffman P., Nicoloff T., *Il manuale MS-DOS per Personal Computer Olivetti*
88 386 0018-X	Hoffman P., Nicoloff T., *Il manuale MS-DOS*
88 386 0184-4	Weber D., *Il manuale Novell Netware*
88 386 0087-2	Jamsa K., *Il manuale OS/2*
88 386 0171-2	Sheldon T., *Il manuale Windows 3 Versione italiana*
88 386 0141-0	Hedtke J.V., *Microsoft Word 4 Guida avanzata*
88 386 0068-6	Matthews M.S., Matthews B.C., *PageMaker 3.0 per personal MS-DOS*
88 386 0146-1	Guerriero M., Zampariolo H., *Programmare in Fred con Framework II e III*
88 386 0033-3	Sand P.A., *Programmazione avanzata in Pascal*
88 386 0051-1	Carroll D.W., *Programmazione in Turbo Pascal*
88 386 0912-8	Carroll D.W., *Programmazione in Turbo Pascal (libro+disco)*
88 386 0060-0	Robinson P.R., *Programmazione in Turbo Prolog*
88 386 0048-1	Peckham H., Ellis Jr W., Lodi E., *Programmazione strutturata in BASIC*
88 386 0070-8	Jamsa K., Nameroff S., *Strumenti di programmazione in Turbo Pascal*
88 386 0919-5	Jamsa K., Nameroff S., *Strumenti di programmazione in Turbo Pascal (libro+disco)*
88 386 0077-5	Mosher F.E., Schneider D.I., *Turbo BASIC Elementi di programmazione*
88 386 0071-6	Schildt H., *Turbo C elementi di programmazione*
88 386 0073-2	Schildt H., *Turbo C Programmazione avanzata*
88 386 0143-7	Schildt H., *Turbo Pascal 4.0 Programmazione avanzata*
88 386 0075-9	Schildt H., *Turbo Pascal Programmazione avanzata*
88 386 0069-4	Schildt H., *Turbo Prolog 1.1 Programmazione avanzata*

QuickBASIC della Microsoft è la versione potenziata e aggiornata del classico BASIC, che è stato per molti anni il linguaggio di prima alfabetizzazione. Rispetto al BASIC standard, è stata aggiunta una serie di caratteristiche che rendono questa versione molto più potente e per alcuni aspetti simile ai linguaggi evoluti quali il Pascal o il C.
Questo volume, aggiornato alla versione 4.5 italiana, descrive in modo chiaro e analitico la sintassi dei comandi di QuickBASIC e, attraverso un ampio numero di esempi, mette a disposizione del lettore le metodologie più appropriate per l'ottimizzazione dei programmi.
La particolare struttura del testo, inoltre, consente sia ai neofiti sia ai programmatori esperti di reperire le informazioni a seconda del livello di conoscenza acquisito.

ISBN 88-386-0172-0

Lire 51.000 (i.i.)

9 788838 601729